WL-TR-91-3107, Vol 1

MODEL ANALYSIS AND EXPERT SYSTEM DEVELOPMENT FOR PLANNING AND SCHEDULING TELEOPERATIONS FOR AIRCRAFT TURNAROUND FUNCTIONS

VOLUME 1 - TECHNICAL REPORT

Dr Eui H. Park
Dr Celestine A. Ntuen

North Carolina A & T State University
Greensboro, North Carolina 27411

Feb 1992
Final Report for Period July 1988 - July 1991

FLIGHT DYNAMICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6553

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

WALTER M. WALTZ, 1stLt
Project Engineer
Special Projects Group

AIVARS V. PETERSONS
Chief, Aircraft Launch & Recovery Branch
Vehicle Subsystems Division

FOR THE COMMANDER

RICHARD E. COLCLOUGH, JR
Chief
Vehicle Subsystems Division

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/FIVM, WPAFB OH 45433-6553 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; Distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S): WL-TR-91-3107, Vol I
6a. NAME OF PERFORMING ORGANIZATION: North Carolina A&T State University; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION: Flight Dynamics Directorate, Wright Laboratory
6c. ADDRESS (City, State, and ZIP Code): Greensboro, North Carolina 27411	7b. ADDRESS (City, State, and ZIP Code): Wright-Patterson AFB OH 45433-6553
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Flight Dynamics Directorate; 8b. OFFICE SYMBOL (If applicable): WL/FIVMB	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F33615-88-C-3400
8c. ADDRESS (City, State, and ZIP Code): Wright-Patterson AFB OH 45433-6553	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. 62201F; PROJECT NO. 2402; TASK NO 01; WORK UNIT ACCESSION NO. 54

11. TITLE (Include Security Classification)
Model Analysis and Expert System Development for Planning and Scheduling Teleoperations for Aircraft Turnaround Functions: Volume I, Technical Report

12. PERSONAL AUTHOR(S)
Eui H. Park, PhD; Celestine A. Ntuen, PhD

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM Jul 88 TO Jul 91	19 Feb 92	97

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Expert Systems, Planning and Scheduling, Knowledge Based Systems

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This report summarizes the results of a 3-year effort to study the planning and scheduling of aircraft turnaround functions using an expert system to minimize the turnaround time. This effort established the feasibility of using expert system technology to model aircraft turnaround functions in a teleoperated environment. The model "opportunistically" generates schedules based on aircraft turnaround functions (refueling, weapons loading, aircraft inspection, maintenance, etc.) and personnel assignments and automation technologies. The model is conceptualized for a dynamic environment. Thus, it is possible for the model to consult its knowledge base in order to identify scheduling strategies to maximize aircraft turnaround efficiency for a particular function.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
Walter M. Waltz, Lt	513-257-2129	WL/FIVMB

DD Form 1473, JUN 86 *Previous editions are obsolete.*

FORWARD

The work described in this report was performed under the guidance of the Special Projects Group, Aircraft Launch and Recovery Branch, Vehicle Subsystems Division, Wright-Patterson Air Force Base, Ohio. Dr. Mangal D. Chawla is the U.S. Air Force Program Manager for this contract. Thanks are due to Capt Dewayne A. Davis who started this project and managed this effort until being transferred. Thanks are also due to Dr. Arnold Mayer, David J. Perez, and Capt Samuel E. Hagins who provided guidance and encouragement from time to time. A special recognition goes to Lt Walter Waltz, whose quick insight and wisdom have been the "knowledge driver" for the model developed for this project.

To our graduate students, Peggy Wang and William Byrd, their tireless work habits and team work are appreciated. The original driving conceptual model provided by Reshetta Roberts is also commended. The authors also thank Ms. Marilyn Riggins for patiently typing this report.

TOP PROGRAM SUPPORT

TOP program environment was developed with the support of the following softwares:

1. NEXPERT™ and NEXPERT OBJECT™ are registered trademarks of Neuron Data Inc., 444 High St., Palo Alto, CA 94301.
2. Artful.lib™ (copr 1990) Artful Applications, Inc. 176 St. George St., Toronto M5R 2M7, Canada.
3. dBASE III PLUS™ copyright (c) Ashton-tote 1984, 1985, 1986
4. Clipper™ (copr 1987) Nantuckett Inc.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS/ACRONYMS

TRSS	-	Teleoperator/Robotic System Simulation.
DAISIE	-	Distributed Artificially Intelligent System for Interfacing with the Environment.
TART	-	Teleoperator and Robotic Testbed.
LART	-	Language-Aided Robotic Teleoperation System.
TOL.O	-	Teleoperator-Oriented Language of the Object-Level.
MSM	-	Master Slave Manipulator.
ATF	-	Aircraft Turnaround Functions.
GPS	-	General Problem Solver.
JC	-	Job Controller.
LD	-	Line Directors.
TOP	-	Task Oriented Planner.
OPS	-	Open Programming System.

CHAPTER 1

INTRODUCTION

1.1 Background

An aircraft turnaround is the process by which the aircraft returning from sortie deliveries are replenished to generate future sorties. Typically, aircraft turnaround functions (ATF) consist of three operations. These operations are A/C servicing, rearming (including reloading ammunition, and reloading bombs and missiles), and minor aircraft maintenance.

The use of robotics technology to perform ATF has been advocated by the Flight Dynamics Directorate of Wright Laboratory [1, 2]. The main motivation for such a concern is crew safety during chemical and biological warfare [3]. The goal to achieve this is to reduce the total man-hours expended in the turnaround operations. Thus, smaller crew size and reduced exposure of the crew to a hostile environment will result in a tolerable risk level.

To assess the feasibility of a robotics program for ATF, baseline concept studies have been performed by Battelle Memorial Institute [3,4]. Some results reported indicate that certain ATF operations can be performed by the robot with minimum human time in the loop.

1.2 The Project Scope

In this project, the effort was to take a deeper look at the use of telerobots for ATF; in particular, the planning and

scheduling of ATF tasks within a limited amount of resources. The following statements of work are used to accomplish the research goal:

- Comprehensive literature review in the area of planning, scheduling, and robotics. A cross-reference index of conceptual and application related works are, therefore, prototyped for telerobotic application in ATF.
- Development of conceptual model for planning and scheduling in the world of ATF.
- Development of a planning and scheduling knowledge base for a well-defined aircraft turnaround function The major accomplishment of this phase of work was on knowledge documentation and requirement analysis for planning and scheduling in the world of ATF.
- Formal development of an expert system for planning and scheduling in the world of ATF and telerobotics. The assumption here is to view a telerobotic system as a multiagent system where agents cooperate to achieve a specific goal.
- The verification and validation of the model were achieved through a direct test with F16 aircraft by the U.S. Air Force personnel supervising the project. By the continuous method of "debug" and "enhance," the planning and scheduling model becomes an open system which allows the user to create plans (or replan) for a generic scheduling scenario.

1.3 A Telerobot System

A telerobotic system consists of the use of robots or general manipulators, and humans for remote operations. Such operations are generally referred to as teleoperations [5].

A teleoperation work environment is an example of a human-machine system. Thus, for such an environment to be useful for what it is intended for, the following characteristics must be available:

1) The system operators or agents must cooperate symbiotically, at least at the highest level of abstraction [6,7].

2) The system must acquire an explicit mode of communication. An explicit communication mode is dialogue based which can provide direct interaction through devices such as a joystick, mouse, visual displays, voice synthesizers etc. [8].

3) The teleoperator must have information from sensors and actuators, perform useful work on its environment, and be controlled remotely by other operators.

These three general characteristics are conceptualized by Sheridan [7] as shown in Figure 1. Ntuen and Park [9] have also presented a general architecture which describes the environment at two levels: functional and operational [see Figure 2].

At the functional level, the telerobotic system requires, a control model for execution of tasks, a supervisory model for issuing directives, a monitoring model for diagnosis and maintenance of system operations, a planning model for managing constraints and scheduling tasks, and a distributed problem solving model for communication between models.

The operational level is the highest level of the teleoperated system. At this level, mental models can be used to represent task scripts abstractly. Issues on the roles the human(s) and machine(s) should play in the symbiont system are addressed conceptually.

As described above, classical modeling tools used in robot control and manipulation fall short of addressing a telerobotic environment. This is so since some degree of human operation is required; the robots act as aids to the operator. Coiffet [10]

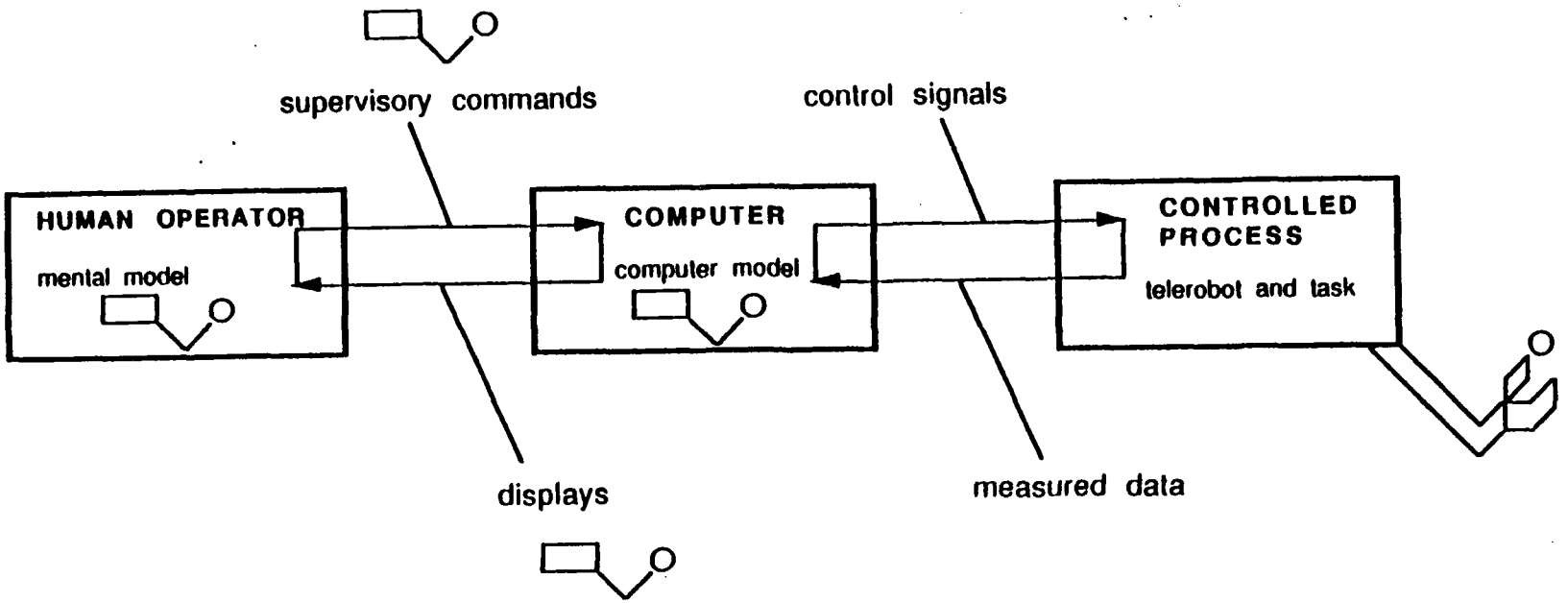


Figure 1: A Conceptualization of the Telerobotic Environment (Adapted from Sheridan: Model of Controlled Process Internal to Human, Computer, Command Language and Display, pp. 32, 1988)

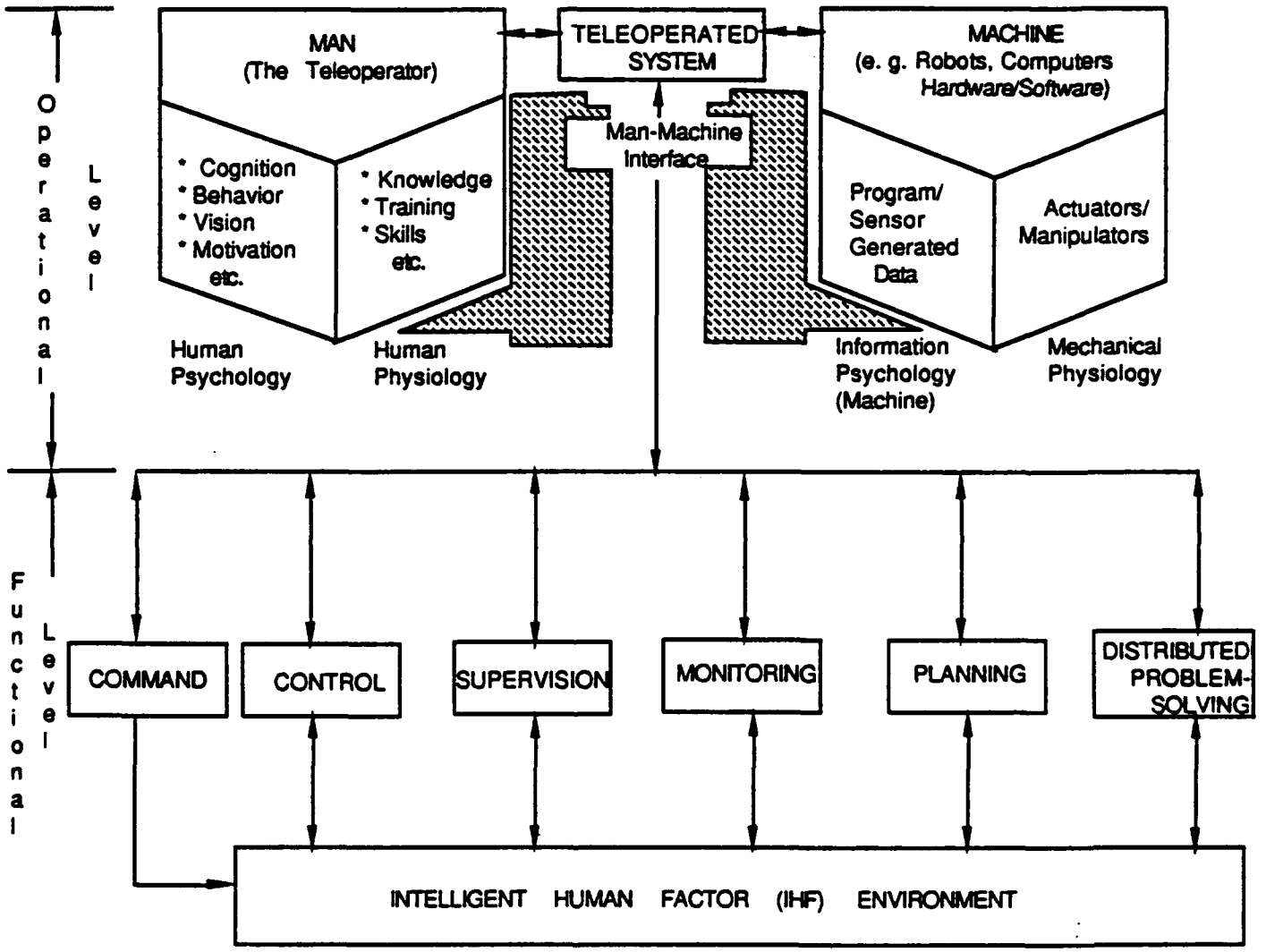


Figure 2: Elements of a Teleoperated System
(Note: • denotes advanced features)

listed the possible areas of modeling complexities as follows:

1. The acquisition and presentation of relevant and easily interpretable information to the operator. Examples of this include the presentation of the stereoscopic view of a gripper, or the indication of the forces existing between two components that are to be fitted together in an assembly process.

2. The automatic monitoring of an operator's movements and the provisions of starting signals which activate the interruption of transmission from master to slave when the precision of the operator is failing. This function is also concerned with the systems self-testing facilities.

3. The automation of various functions so freeing the operator. These might include the gripping of objects when a device is in an automatic mode, or the maintenance of the horizontal when grippers are used to handle fluids, irrespective of the motion associated with their handling.

Since all three problems cannot be solved in a single model, we have chosen to address problems 1 and 3. Our approach is to utilize rule-based expert system technologies for planning and scheduling in a telerobot environment. We recognize that using an expert system will compensate for the most difficult problems associated with the mathematics of robotic control. Our expert system is generic and allows for generalizations in the world of domain-specific planning.

1.4 Outline of Report

This project report is organized as follows:

- Chapter 2 presents a literature survey on planning and scheduling techniques for possible technology transfer into telerobotic applications and aircraft turnaround functions (ATF).

- Chapter 3 presents the development of a conceptual model for planning in a telerobotic system. The concept presented is both generic and open-ended for understanding knowledge requirements in modeling teleoperation.

- Chapter 4 presents a planning paradigm in the world of ATF. The methodology lies primarily on task consideration and resource needs in planning a multiagent system.

- Chapter 5 presents a knowledge-based model for scheduling resources in an ATF domain. A heuristic algorithm that employs results from the planning is discussed.

- Chapter 6 presents some sample applications of the planning and scheduling models in aircraft turnaround functions for F16A aircraft.

- Chapter 7 concludes the project report with a summary of what has been accomplished and a discussion of the directions for future work in planning multiagent systems for ATF domain.

CHAPTER 2

LITERATURE REVIEW IN PLANNING, SCHEDULING, AND ROBOTICS WITH APPLICATIONS TO AIRCRAFT TURNAROUND FUNCTIONS TELEOPERATION

2.1 Introduction To Literature Review

One of the tasks for this project was to conduct a literature review so as to understand the relationship between planning and scheduling concepts and how they could be used in telerobotics, and recommend appropriate planning and scheduling techniques for aircraft turnaround functions.

2.2 Planning Concepts and Their Brief Histories

The term "planning" has gained significant meaning in describing problem-solving protocols in the AI community. Although the term is as old as human existence, its application in the development of problem-solving systems has given entirely different meanings to the concept.

Generally, planning methods require one to compile some qualifying information about a given problem domain [7,11,12,13]. From this, pieces of information which are likely to guarantee useful contributions toward solving the problem are retrieved and organized for that purpose. We refer to this kind of planning as "naive" planning.

The artificial intelligence-based definition of planning can be described more succinctly as the systematic vis-a-vis experimental compilation, organization, and use of domain knowledge for the purpose of automatic solution finding to a given problem via the computer [14,15]. The AI community has, in the past 20

years been involved in the development of such systems. These are referred to by such names as "planners," "problem-solvers" or "expert systems advisors."

The concept of planning for intelligent problem-solving systems can be traced to Newell, Shaw and Simon [16] in developing a planner called General Problem Solver (GPS). They introduced the "mean-ends analysis" paradigm which solves problems by applying an operator that would achieve some goal of the problem and taking the preconditions of the operator as new goals. STRIPS [17], due to Fikes and Nilsson, modified the GPS concept to that of an action model - in which steps have post-conditions that are the only things that get changed by the step. These two planners, GPS and STRIPS, operated on nonconjunctive tasks in an elementary "block-world" domain.

The growing development in the field of planning research has led to different representational view points, constructs and implementation concerns [13]. These concerns led to the special workshop in planning held in Santa Cruz as sponsored by DARPA [18]. The workshop revealed the following types of planning taxonomies;

1. A Tactical Planner: a planner who is primarily concerned with deciding what to do in situations in which available information is limited or uncertain.

2. A Linear Planner: In STRIPS [17] for example, actions are represented as functions from sets of sentences to sets of sentences in some appropriate language. Such a representation allows for what is disparagingly referred to as linear planning. In a linear planning framework, plans correspond to sequences of primitive actions.

3. A Hierarchical Planner: This is perhaps the best known and most misused technique since a number of unclarified concepts get tangled with its application. Hierarchical planning arose

out of dissatisfaction with linear planning. A hierarchical plan starts at the general level of planning abstraction and moves down to specifics, details, subplans and levels. Problem-solving with hierarchical networks occurs via conflict resolution [1], and interactions between subplans.

4. An Opportunistic Planner: This is a planning system whose actions co-routine with the model environment to dynamically instantiate or alter a problem-solving behavior based on circumstances. That is, during each point of a problem-solving life cycle, the planner's current decisions and observations suggest various opportunities for further plan development and changes. Originally suggested by Hayes-Roth and Hayes-Roth [19], the concept has been used to explore opportunistic scheduling of manufacturing systems [21,22,23].

5. A Least Commitment Planner: *NOAH* [7,23] and his descendants operate on the premise that operations are not to be sequenced unless absolutely necessary. A set of procedures known as critique agents are used to detect and correct interaction, eliminate redundant operations, and so forth.

6. Case-Based Planner: Case-based [24,25,26,27] planning systems take as a starting premise that the organization of experience is paramount in formulating new plans and debugging old ones. Case-based reasoning is a simple idea: solve new problems by adapting solutions known to work for old problems. The Case-Based Planner offers several potential advantages over rule-based reasoning systems: rules are not combined blindly in a search for solutions, solutions that are generated can be explained in terms of concrete examples, and performance can improve automatically as new problems are solved and added to the case library.

7. Agenda-Based Planner: This is a class of planning systems using procedural listing of objects, events, and activity occurrences during the problem-solving life cycle. Heuristic models, such as priority rankings, utility preferences, or economic values are used to "promote" or "demote" plan structures in the agenda list. Example of a planner in this category is SIPE [28].

8. Endorsement-Based Planner: This is a planner which develops and refines plans based on the user's actions and possible intentions. The concept of "intention" allows the planner to reason from the human behavior perspective [See, e.g.; 35-37].

A summary of some planners reviewed is shown in Table 1.

TABLE 1: CLASSIFICATION OF AI PLANNERS

PLANNER/YEAR AUTHORS	FUNCTION	APPLICATION DOMAIN	PLANNING PARADIGM	PLAN REPRESENTATION	TYPES OF CONSTRAINTS & RESOLUTION PROCEDURE	SPECIAL FEATURES & COMMENTS
BUILD/1974 Fahlman [16]	Assembly Tasks	Building construction	Nonhierarchical	·Pseudo-associate pattern data base ·List structural	Plan-phase geometry	-Calliable functions from any source - Backup plan - Integral Cooperation
PULP-I/1979 Tangwongsan and Fu [65]	Scheduling robot tasks with human off-line control	·Pseudo-Telerobotics ·Intelligent control systems	Nonhierarchical	·Procedural knowledge base ·Semantic nets	·Task specific ·Pattern matching ·distance metrics	Supervised learning system
FPS/1985 Sobek [61]	Robot plan generation and execution tasks	Robotics	Nonhierarchical	Rule-based production systems	Domain specific constraints	Dynamic plans
TWEAK/1987 Chapman [8]	General paradigm for planing	Block World and Robotics	-Hierarchical opportunistic	·Predicate calculus	·Constraint posting -provability of conjunctive relationship -codesignation of constrints	-sufficiency conditions for goalachievement, protection, and instatiations

TABLE 1: CLASSIFICATION OF AI PLANNERS (cont)

PLANNER/YEAR AUTHORS	FUNCTION	APPLICATION DOMAIN	PLANNING PARADIGM	PLAN REPRESENTATION	TYPES OF CONSTRAINTS & RESOLUTION PROCEDURE	SPECIAL FEATURES & COMMENTS
GPS/1960 Newell, Shaw and Simon [44]	·Games ·Theorem Proving	·General problem solving environment ·Experimental environment for object programming	Hierarchical	·Predicate calculus ·State-space	·Task defined: means end-analysis	·First AI-based planning system ·Nonconjunctive planner
STRIPS/1971 Fikes and Nilsson [17,18,20]	Assembly Tasks	Robotics	Hierarchical	-Procedural Nets -Declarative production system Propositional logic	·Task specific -GPS means ends-analysis -Resolution by theorem proving	·Learning process by constructing macro-operators ·Domain-independent conjunctive planner
ABSTRIPS/ 1974 Sacerdoti [54]	Robotic Task scheduling (extension of STRIPS)	Robotics	Hierarchical	·Function Schema ·FOL	Task specific -Critically factor	·Plan refinement procedures procedure in non-linear fashion
NUDGE/1977 Goldstein and Roberts [26]	Office scheduling e.g. scheduling meeting between executives	Scheduling Conflicting goals	Hierarchical	Frames	·Preference constraints ·Causal constraints -Preference relaxation heuristics · Priority ordering	Possesses a domain-independent system called BARGAIN which can be detached to solve similar search problems in scheduling

TABLE 1: CLASSIFICATION OF AI PLANNERS (cont)

PLANNER/YEAR AUTHORS	FUNCTION	APPLICATION DOMAIN	PLANNING PARADIGM	PLAN REPRESENTATION	TYPES OF CONSTRAINTS & RESOLUTION PROCEDURE	SPECIAL FEATURES & COMMENTS
PIGS/1984 Davis and Comacho [13]	Pathfinding	Robotics	Nonhierar-chical	·Predicate calculus	Task specific	Extension WARPLAN
LAWALY/1973 Siklossy and Dreussi [59]	Assembly Tasks and general plan iden-tification in micro-world	Robotics	Hierar-chical	·Procedural knowledge base ·Object list data base	·Task specific -subgoal re-ordering using dis-approval proofs	Accelerated search procedure - Procedural learning
FIXER/1985 Grant [27]	Maintenance planning and schedul-ing	Aircraft	Hierar-chical	Relational Production Systems	Organiza-tional, Causal, Pre-ferences and availability constraints -Heuristics -Order list-ing and priority selection rules	Advice-giving system
DEVISER/1983 Vere [68]	General purpose automated Planner/ Scheduler in micro-world	Space craft scheduling and simula-tion	Hierar-chical	Relational productional systems	·Resource, organiza-tional, task, time-constraints -Least Commitment -Event occurrences -Queue dis-ciplines	-Generates par-rallel plans -Handles tempo-ral variables -Provide as win-dows user "work bench" -Dynamic en-vironment -Discrete event simulation

TABLE 1: CLASSIFICATION OF AI PLANNERS (cont)

PLANNER/YEAR AUTHORS	FUNCTION	APPLICATION DOMAIN	PLANNING PARADIGM	PLAN REPRESENTATION	TYPES OF CONSTRAINTS & RESOLUTION PROCEDURE	SPECIAL FEATURES & COMMENTS
NOAH/1975 Sacerdoti [74,75]	Assembly Tasks	Robotics	Hierar-chical	-Procedural Nets -Declarative production system Pro-positional logic	·Task specific -Least commitment using critic functions	·Meta-rules ·Statistical learning ·Recallable program
MOLGEN/1980 Stefik [62,63]	Assist in planning molecular genetic experiments	Biochemistry Portable for other envi-roment	Hierar-chical	·Production Systems	·Causal con-straints -Layered guessing -Least co-mmitment -Constraint refinement and pro-jection -Difference reduction	·Meta-rules ·Statistical learning ·Recallable pro-gram
SIPE/1984 Wilkins [72]	Cooperative and Parallel Planning of general problem solving system	General problem Ro-botics and Assembly Tasks	Hierar-chical	Distributed knowledge base · Production Systems	·Causal con-straints	Useful in co-operating/ parallel plann-ing applications

TABLE 1: CLASSIFICATION OF AI PLANNERS (cont)

PLANNER/YEAR AUTHORS	FUNCTION	APPLICATION DOMAIN	PLANNING PARADIGM	PLAN REPRESENTATION	TYPES OF CONSTRAINTS & RESOLUTION PROCEDURE	SPECIAL FEATURES & COMMENTS
TOUR/1977 Kuipers [40]	Routing	Logistics and scheduling	Hierar-chical	·Frames ·Topological network (re-presented by procedures)	Task specific -Qualitative simulation -Scenario mapping ·Environmen-tal	Task cognition and learning

2.2.1 Evaluating Planners For Telerobotics Applications

Successful implementation of computer based systems is usually need-dependent. Therefore, in order to buy or develop a planning shell for a problem-solving system, several issues have to be resolved. We present the basic and most obvious technical issues:

1. **Language** -- A plan must first have a vocabulary of symbols and notations in which the initial state, goal conditions, and operators may be represented with conceptual objects. For example, the assertion: INROOM (ROBOT, ROOMA) means that the mechanical device called ROBOT is in room called ROOMA. Some languages have been developed based on the planning environment [29].
2. **Intention** -- A plan must have a purpose or intention. This characteristic allows a planner to act in a directed, domain-specific fashion. It is possible for a plan to have multi-intentions, and subplans should be developed in a layered fashion to handle such intentions [27].
3. **Belief System** -- A plan must contain some specification or instantiation of beliefs about the environment for which the plan is designed [30]. The availability of an embedded belief system in a planner allows for on-line result validation and verification.
4. **Conflict Resolution Capability** -- A planner should be able to trouble-shoot the problem environment as well as understand and resolve basic conflicts.

5. **Cooperation** -- A planner should be able to incorporate knowledge from other (user-defined) plans toward solving a common problem. This principle is known as "Cooperative Planning" [15,31].
6. **Authority** -- A planner should possess an authority to determine (using its knowledge base) whether to exclude one or more subplans during execution of a problem.
7. **Responsiveness** -- A planner should respond to events occurring during the plan execution. The number and magnitude of changes may be sources of difficulty. This concept is known as "Replanning Ability" [32,33].
8. **Plan length and predictability** -- The more predictable the execution environment, the longer the plan can be with a reasonable expectation for successful completion. Automatic programming, in particular, can produce plans (programs) millions of steps long that usually complete successfully. Producing such big plans is only possible because a computer is such a predictable environment; the main source of unpredictability here concerns the input to the program.
9. **Correctness versus robustness** -- Traditional AI planning systems tend to concentrate on producing correct plans. Although this method is appropriate for highly predictable environments, it is much more important to produce robust plans in realistic situations. Robustness means the plan is likely to succeed no matter what unanticipated conditions arise; that is, robust plans avoid including commitments to

courses of action which allow few options if they fail in execution [34,35,36].

2.3 Scheduling and Task Assignment

Most scheduling and task assignment models can be classified into two general approaches: quantitative and qualitative. Discussions on these issues are presented below:

2.3.1 Quantitative Approach To Task Assignment Problems

The quantitative approach can further be classified into operations research and control theoretical models respectively. Operations research models for task assignments are based on scheduling theory. Scheduling problems arise in many partial circumstances and under a wide variety of guises [See, e.g., 21,22,25]. Many are basically optimization problems having the following form: given a collection of tasks to be scheduled on a particular processing system, subject to various constraints, find a feasible schedule that minimizes (or in some cases maximizes) the value of a given objective function. Thus, it is not surprising that much of the work on scheduling theory has been devoted to the design and analysis of optimization algorithms---algorithms that, when given a particular instance of a scheduling problem, construct an optimal feasible schedule for that instance.

Unfortunately, although it is not difficult to design optimization algorithms (e.g., exhaustive search is usually applicable), the goal of designing efficient optimization

algorithms has proved much more difficult to attain. In fact, all but a few schedule-optimization problems are considered insoluble except for small or specially structured problem instances. For these scheduling problems, no efficient optimization algorithm has yet been found and, indeed, none is expected.

For these reasons practitioners often are willing to settle merely for a feasible schedule, so long as they have some indication that it is a reasonably good one [37]. Though there are problems for which finding even a feasible schedule seems computationally hopeless, for most schedule-optimization problems there do exist simple heuristic algorithms that find feasible schedules quickly.

At this point it is convenient to introduce a bit more formalism into our discussion. A scheduling problem will consist of two parts: a model and an objective function [22,23,37]. The model describes the system, including the kinds of tasks and the constraints on their processing, the types of processors and their number, and all other properties necessary to specify feasible schedules. The objective function assigns a "value" to each feasible schedule. An instance of such a problem is merely a specification of a particular system, set of tasks, and set of constraints conforming to the model. Given an instance I, the model tells us what the feasible schedules for I look like and the objective function tells us what their values are. The optimal schedule value for I, which we denote by OPT(I) is the minimum (or

in some cases the maximum) of the values for all feasible schedules.

One scheduling model that fits many task assignment problems is the following: The system is a set of m identical processors, $\{P_1, \ldots, P_m\}$ that operate in parallel. The set $T=\{T_1, \ldots, T_n\}$ of the tasks to be executed consists of independent tasks (no ordering constraints between them, each of which has an execution time $\tau(T_i)$ and requires only one processor).

The only processing constraints are that no processor can execute more than one task at a time and that, once a processor begins executing a task T_i, it continues executing T_i until its completion $\tau(T_i)$ time units.

Although this description of the model has been in abstract mathematical terms, the model does correspond to a simplified version of many real task assignment systems. Examples are a computer center containing a number of computers or even a typing pool, where the typists are the processors and the letters and papers to be typed are the tasks.

In such a system one desirable goal might be to get all the work done as soon as possible. We shall be interested primarily in the objective function corresponding to this goal, called the finishing (turnaround) time of a schedule. The finishing time for a schedule is the earliest time at which all tasks have been completed.

This problem of finding a feasible schedule with minimum finishing time is known as the independent task-scheduling problem.

The mathematical programming procedure is formulated as:

Given $R = (r_1, r_2, \ldots, r_n)$

Min $Z = Z(r_1, r_2, \ldots, r_n)$

S.t.: $r_1 + r_2 \ldots + r_n \leq R$

The r vectors could be training, equipment design, selection and job difficulty.

Another quantitative model similar to the mathematical programming model (MP) is the control theory. In control theory terms, it is assumed that the human acts as a servomechanism. Using this analogy, the allocation and task assignment problems are then conceptualized and modeled as feedback control systems. The objective function of interest is the "gain" realized by cooperative task [33,38].

2.3.2. Qualitative Approach To Task Assignment and Scheduling Problems

An assignment problem is used where job rating and the operator have to be assessed using some criteria. For example, an operator may be rated in terms of physical effort, tasks demands, level of attention span, concept of stress - i.e., the time required to execute a series of tasks weighted by the time available to perform the task, and/or the level of stimulus dimensions [39,40].

Another qualitative technique that is gaining attention recently is fuzzy analysis. The theory of fuzzy systems seems to be suitable for tasks rating since it allows one to interpret and

manipulate imprecise information, and incorporate human judgement in order to improve the overall analysis.

2.4 A Review of Work In Telerobotics

The framework of research works in telerobotic systems can be summarized by Sheridan's [41] metaphorical statement:

> "A telerobot is a teleoperator which also embodies understanding memory and decision capability so that the human operator, as a supervisor, may communicate to its high-level goals and contingencies and receive high-level state information, while the machine executes low level functions and pieces of the task semi-autonomously by closing the loop through its own effectors, sensors and internal computer."

The above observation presents the stratification in research methodologies for telerobotic environments. In summary, research in telerobotics has evolved in at least five directions as follows:

2.4.1 Control

This involves changing the teleoperator actions according to some emerging plans. A survey in this direction can be seen in Yang et al. [42,43].

2.4.2 Supervision

A supervisory control system is basically a feedback system with the capability to monitor the actual operating state of the system and to keep it within the specified target domain, to coordinate disjunctive efforts [44], to supervise learning such as using a joystick to train the robot, and to manage strategic

decisions such as giving directives and overriding policies or priorities.

2.4.3. Distributed Problem Solving

Distributed problem solving is an issue currently being addressed in telerobotic system research. Distributed problem solving is concerned with hierarchical and parallel problem solving at the system level using global models of abstractions. The idea is that if several computers or teleoperators can be delegated to do tasks which they can do best, then a significant amount of problem solving time can be realized.

2.4.4 Monitoring

Monitoring a symbiont system is more difficult than monitoring an ordinary signal-agent system. This can be explained from the fact that each agent has behavior which may be significantly different from those of the other agents or the overall systems goal(s) and intention(s). Research suggestions and directions in system monitoring have focused primarily in the area of real-time observation [45], fault diagnosis and inspection, parameter evaluation and estimation and performance auditing [46].

2.4.5 Communication

Communication research in a human-machine system seems to emphasize a mixture of implicit and explicit models of communication. Explicit communication is a dialogue-based

communication that requires the human to communicate with the task allocator using an input device such as a keyboard, mouse, or lightpen, or by using his voice, buttons, or switches. Although this type of communication has the advantage of minimizing misunderstanding in intent between the human and the task of executor, it is unfortunately costly in terms of taking up more of the human's time due to the human having to stop performing tasks to communicate with the task allocator.

Implicit communication is a model-based communication in which the computer uses models of the human to predict what the human is likely to do next. From this prediction, the computer attends to tasks which are likely to be neglected by the human. Implicit communication is typically used when the human performs the majority of the tasks. This type of communication has the advantage of allowing the human to execute tasks without having to communicate with the task allocator. The disadvantage of this method is that it requires development performance which is usually difficult to build and results in an imperfect model of the human.

2.5 Prototype Systems Developed For Telerobotic Applications

Several test-bed systems have evolved since the 1980's which are dedicated to teleoperated functions. Among these are the following:

2.5.1 TRSS - Teleoperator/Robotic System Simulation

TRSS [47,48] is a modular software simulation coupled to a reconfigurable teleoperator control station. The module resides in

a relatively powerful processor that can coordinate communication from other processors and devices. The module deals with "strategic" task planning, database management of the machines concept of the environment, supervisory monitoring of the teleoperator control station, and the interfaces to various "tactical" controllers.

2.5.2 DAISIE - Distributed Artificially Intelligent System for Interfacing with the Environment [49]

This is implemented at NASA-Langley as a system that divides the control of a sensor/cognition/actuator system into two major hierarchical levels. These levels are termed strategic and tactical. The "tactical" level refers to local, specific control of a particular sensor/actuator grouping (preceptor/proprioceptive actuator). "Strategic" refers to a control level with a global view of all tactile units and their actions. DAISIE implements the concepts of using various degrees of abstraction at different goal levels.

The DAISIE system exploits distributed processing within the limitations of the available hardware. Functions such as manipulation, vision, end-effector control, and force-torque sensing are each run by separate processors. The higher "intelligent" levels are also distributed in separate processors.

2.5.3 TART (Teleoperator and Robotic Testbed)

Harrison and Orlando [48] discussed the use of TART

implemented on the VAX 11-750 in the ISRL (Intelligent System Research Laboratory). TART is a layered driven model in which each successive layer provides additional value to the system. Currently, five layers are implemented: 1) user, 2) system, 3) scheduling, 4) communication and 5) servo/sensor. The lowest four layers of TART are designed for error-checking required by user applications. Users are encouraged to use only the TART-defined system level mechanisms for modification of data structures.

2.5.4 LARTS (Language-Aided Robotic Teleoperation Systems)

LARTS [50,51] incorporates two sets of teleoperational languages with a master-slave manipulator. Both spatial and temporal autonomy which support the operator are provided by the languages. The authors of LARTS focus on the structuredness in teleoperational task execution. For example, there may exist many constrained motions in the handling of objects. Elementary tasks, such as pick, place, remove, grasp and so on which are executed repeatedly are examples of teleoperational task execution.

2.5.5 TOL.O (Teleoperator-Oriented Language of the Object-Level)

To cope with the burden of the operator having to teach the actual environmental data in the program, a special teaching method designed for teleoperation shows a synopsis of the method [50]. TOL.O [52] is designed to specify elementary task motions of teleoperation. TOL.O instruction yields the program for the task. The programming burden for the operator is, therefore, reduced.

Specifications in TOL.O are as follows:

1) Operator declares aim to task using TOL.O instruction.

2) Instruction is automatically expanded into the motion-level task procedures.

3) Teaching-executing systems interpret the draft program steps one by one and prompt the operator to do the necessary motions for the task execution and teaching.

4) Operator executes the task by operating the masterslave manipulator and signals the system, using a button, that the motion is completed.

5) At the end of task execution a program consisting of motion procedures together with the environmental data is stored in the system.

2.5.6 MSM (Master-Slave Manipulator)

The MSM [51] language describes the software jigs and control schemes of the man-slave manipulator. The instructors related to the software-jig fall into the following three categories:

1) instruction specifying elementary motion;
2) instruction which construct the jig body by combining the constraints;
3) instructions describing the attachment and detachment of the jig body.

2.6 Summary

The review of literature presented here was to abstract information related to planning, scheduling and telerobotics and how they could be used in aircraft turnaround functions. The literature available in these areas is so large that we cannot present all of it here. However, we have attempted to review the basic facts related to the world of aircraft turnaround functions. These reviews have indicated the following observations.

1) Task assignments have been unitary and sequential. In a telerobotic system, multitask assignment problems have to be addressed. In sequential task assignments, the strategy seems to assign a fixed subset of the tasks to each resource prior to job execution. The basic problem in this type of assignment is fault intolerance, i.e., if one resource failed in performing its task, another resource could not take over the operation of the task. A more flexible and dynamic assignment is proposed in this project. In the dynamic task assignment, any resource which is currently free and capable to perform a task could be assigned the next task to be performed.

2) Planning and scheduling profiles discussed are deterministic. In deterministic scheduling, all jobs, resources, and task completion times are known with certainty. The symbiont behavior or a telerobotic system seems to invalidate any deterministic assumptions about the system parameters.

3) In the literature, most of the task allocation problems seem to consider structural type constraints. However, in addition to the structural constraint each teleoperator has a skill inventory that may significantly constrain the assignment problem. In this project, a more global view about constraints is used and assignments are based on both tne satisfaction of such constraints and resolving conflicts between the teleoperator capabilities.

CHAPTER 3

THE DEVELOPMENT OF A CONCEPTUAL MODEL FOR PLANNING IN A TELEROBOTIC SYSTEM

3.1 Introduction

The aim of this chapter is to discuss a conceptual model required for planning a telerobotic system. As indicated in a previous study by Orlando [49], the complexity of a telerobotic system is such that "interleaving of the steps of plan creation and plan execution must be conceptualized at the highest level of abstraction prior to modeling." We show in this chapter that planning in a telerobotic system requires an understanding of a human-machine (robot) working together symbiotically. Among several other things, this can take place at four dimensions: (1) understanding the role of humans in teleoperated tasks (Expert system), (2) understanding the task environment and knowledge required to accomplish the task (Knowledge-Based System), (3) understanding the communication and problem-solving approaches for a "symbiotic" system (Distributed Problem-Solving Environment), and (4) understanding the cognitive requirements that allow the agents (robots, human, computers, etc.) to learn from the accomplishment of one another (Learning System). This concept is shown in Figure 3 and it defines the components of a telerobotic planner.

3.2 The Expert System: Understanding the Role of Humans in Teleoperated Tasks

The use of robot technology along with human labor does not

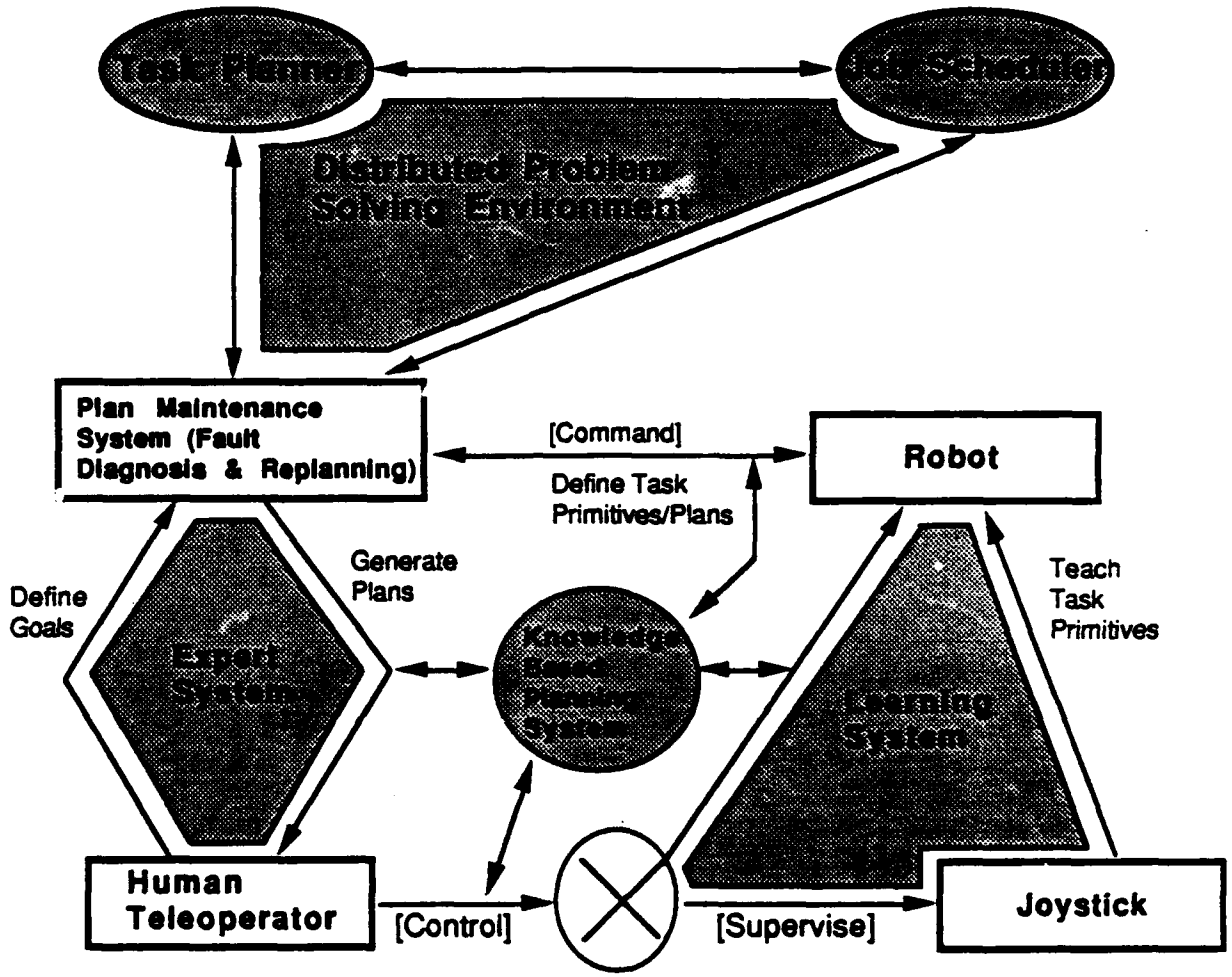


Figure 3: A Conceptual Model Environment For Planning And Scheduling In A Telerobotic System.

necessarily remove humans from the system in which a task is to be performed. As noted by Rasmussen, "... basically it moves them from the immediate control of system operation to higher-level supervisory tasks and to long-term maintenance planning tasks."

Because of the expected changes in the human role as a controller to supervisor, the control expertise must be preserved and transferred to the robot. The realization of such transfer can be achieved via expert system technologies.

Expert systems are computer-based models that can solve problems that are normally solved by the human "experts." For a telerobotic system, the roles of an expert system module can be to:

- direct both the robot and human to goal attainment;
- provide multiple context advice;
- supervise the human and the robot in task allocations;
- provide suggestions and alternatives to problem solving situations.

3.3 The Knowledge-Based System: Understanding the Task Environment and Knowledge Required to Accomplish Tasks

To solve expert-level problems in which several agents interact, an access to a substantial knowledge base is required. In fact such a knowledge base should be dynamic with response to different task environments. Generally, such knowledge occurs at two levels, teleological knowledge and epistomological knowledge, respectively.

Teleological knowledge relates to the entire design spectrum of the teleoperated system. The morphology of the design can be understood only through the analysis of human and machine (robotic) capabilities with respect to task requirements.

This concept is referred to as "mixed initiative" [40]. "The mixed initiative concept is based on the transfer of authority and control responsibility between an automated system and a human operator." Thus, the knowledge (information) for planning relates to task, control, and feedback.

Examples of teleological knowledge are robot control algorithms, human discrimination of occluded environments, or cognitive skills required for a particular task (domain knowledge).

Epistomological knowledge relates to the information and data required for understanding a task situation defined at the highest level of abstraction. An introduction to epistomological constructs for teleoperated systems has been discussed by Orlando [49]. In her discussion, an evolutionary approach where knowledge is activated, complied, managed, and controlled from the bottom-up is presented. The knowledge elements identified for modeling teleoperations are as follows:

1. Intrinsic and extrinsic data compilation. This involves a process of deep generation of information to describe a system's behavior. Psychologists refer to this as a pseudo bio-feedback information processing [53].

2. Thematic and rhetorical knowledge of the system. This involves some sense of spatial cognition where information about a situation is stored in the form of dynamic production rules [54].

3. Introspective Knowledge. This involves data/information based on reflective assimilation of the environment. In the telerobotic system, a simulated perceptual representation of a picture by computer vision can provide informal introspective data.

4. Perceptive Knowledge. This involves data based on physical sensation as interpreted in the light of experience. Orlando [47] used a theory of physiological psychology to explain the "hierarchy of abstraction" of knowledge evolution. Perceptive knowledge for robotic software systems requires instantaneous

cognition of "foreign" objects that may pose a threat to mission participants.

5. Catalog of Intentions and Meanings about Concepts. These relate to data used to describe the purpose of the (planning) system and expected goals. The data or constructs can be ill-structured, fuzzily described and/or possess temporal attributes.

3.4 Distributed Problem-Solving Environment

A telerobotic system requires that humans and artificial agents (telerobots and computers) cooperate in decision-making and control of tasks in a complex, unstructured, and dynamic environment. Unlike the manufacturing robots, telerobots are expected to possess "exceptional" intelligence, be flexible, and exhibit a high level of dexterity and reliability. The existence of these skills and requirements represent "virtual" attributes which must match those of the human operator -- at least conceptually if the true meaning of "symbiosis" is to be modeled into cooperative and distributive planning. Interests in telerobotics require work on cooperative problem solving. In this environment, we conceive of a group of teleoperators who form a "crew" to perform a particular task known as action units. Many modeling issues of concern for effective cooperative problem solving include:

(1) Agents should cooperate during task execution. Thus, task assignment based on subtask decomposition should rely implicitly on the agent's capability.

(2) A distributed (and cooperative) problem solving environment should possess at least one agent who can "authorize" the execution of plans. Thus, the idea of priority is crucial in such an environment.

(3) Agents need to represent and reason about their own actions, the actions of other agents, and their interactions between

agents.

(4) Agents need to coordinate their interactions and possess interleaving plans so that they work as a coherent team when cooperating to achieve a shared goal.

3.5 The Learning System: Understanding the Cognitive Requirements in a Telerobotic System

The capabilities of "symbiotic" system agents to learn from one another is an issue of concern in modeling a telerobotic environment. There is significant support in the literature by cognitive theorists [55,56] that suggests the need for learning as a paradigm in which a human-machine environment can work cooperatively. Because many tasks are difficult to perform jointly in a telerobotic system, especially if communication time delays exist, an appropriate learning mechanism is needed. Such a learning mechanism may involve human intelligence (knowledge), sensory and robotic capability to perform remote manipulation tasks, etc. A particular consideration of learning in a telerobotic system is skill acquisition by robots [57].

Skilled tasks are multicomponential and heterogenous in nature, requiring mixtures of cognitive, motor and perceptual abilities [58]. Acquisition through the learning of performance strategies is typically necessary when tele-autonomous systems are desired. Thus, for a telerobotic application, a learning system should interact between the human teleoperator and the robot such that:

(1) The human can teach basic task primitives to the robot.

(2) The robot can learn what is being taught through cognitively driven models.

(3) The robot can learn about task environments, generate plans, and respond to schedule or scenario changes.

3.6 SUMMARY

A teleoperation requires that both human and artificial agents (robots) cooperate in decision making during task execution. Task planning in such a system requires more than the classical modeling techniques because of interactions of several nonquantifiable parameters. One approach to modeling is, therefore, first to conceptualize the levels of abstraction, and then construct a domain-specific simulation of the task environment. We have identified and discussed these levels of abstractions, the roles of human (experts), the knowledge base requirement, the distributed method of problem solving that integrates the human and machine (robot) capabilities, and the need for skill acquisition models between agents using the concepts from learning literature.

CHAPTER 4

PLANNING IN THE WORLD OF ATF

4.1 Introduction

A teleoperation involves multiagents which are supposed to interact symbiotically during task execution. The interdependence of these agents comes about from sharing the same limited resources. Particularly, each of the agents has its own state control variable and goal functions. Therefore, in planning a teleoperation, the total amount of the resources available becomes a decision variable itself. Hence, teleoperation is a constraint-directed planning problem similar to open job shops as observed by Fox [20].

In addition to the above observations, planning teleoperation is also constrained by its unstructured domain. In fact there are at least four scenarios that contribute to this unstructured environment. Some of these are:

(1) Planning teleoperation is context-based; therefore plan decisions are made contextually at each level of abstraction.

(2) The basis and criteria for optimal planning and scheduling policies are not well defined since the environment is usually unstructured. Here, the central issue is not to optimize a given schedule as in manufacturing systems.

(3) There are induced lags or time delays in information flow between the agents. Thus, a plan change during a time lag may change an entire schedule. Of course, replanning problems cannot be solved analytically except by a rich domain independent knowledge-based system.

4.2 Plan Synthesis for ATF

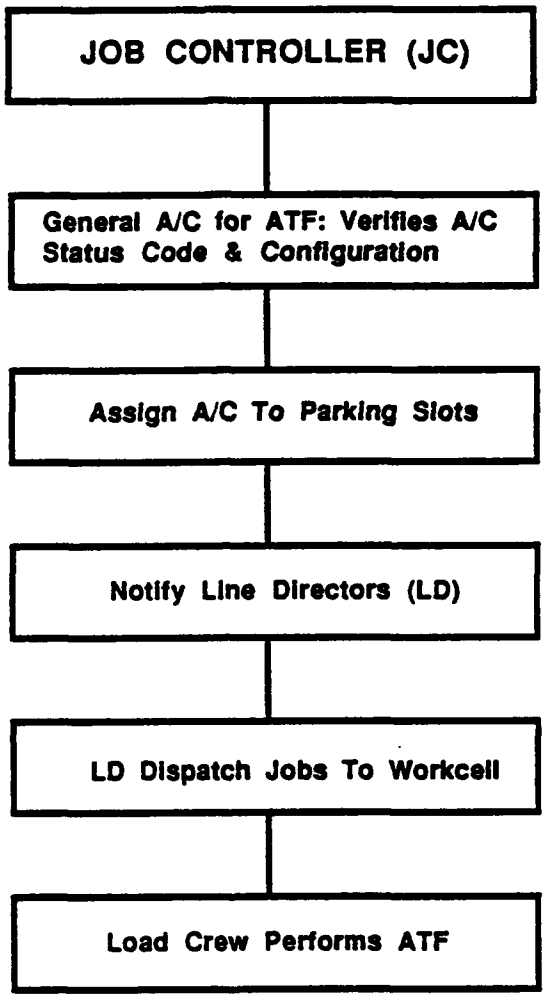
In the aircraft turnaround scheduling domain, many actions are reactions to a current situation. Ordinarily, a human can perform actions in order to change his future in a desirable way. Habits and memorized behavior (such as identification of aircraft payload configuration) are examples of these actions. Not normally regarded as problem solving per se, they are, nevertheless, plans stored for certain primitive actions such as aircraft refueling.

When a nonhuman agent is considered, such a plan of action requires high-level cognitive tasks. For example, the capability of a robot program to recognize and integrate predictive behaviors in a real-time problem solving scenario: looking ahead to predict potential collisions and events which require interventions.

This scenario is applicable to an ATF domain which can be described as follow:

> The ATF tasks consist of servicing, rearming and minor aircraft maintenance during combat missions or training exercises. When an aircraft is returning after delivering a sortie, the pilot relays the status of the aircraft to the ground crew in order to give them a head start on preparations for turnaround. The crew chief makes a thorough visual examination of the aircraft based on the pilot's information. If the aircraft has no damage, it is taxied to the turnaround area where the turnaround functions are executed. This description can be shown in Figure 4.

The A/C Turn Director generates a plan based on his particular needs (context). Usually, these plans will consist of the three functions of aircraft turnaround. Each of these functions can be performed in some locations (known as stations) in the aircraft. Figure 1 shows example stations for ATF.



Acronyms

ATF = A/C Turnaround Function

JC = Job Controller

LD = Line Directors

Figure 4: A General Information Flow Scenario for ATF

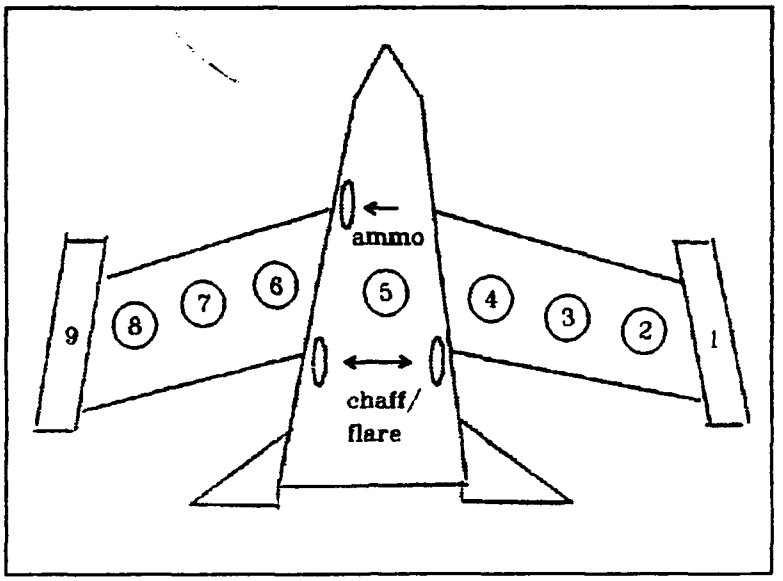


Figure 1: Aircraft Work Stations

Typically, a plan is a selection of one or several configuration codes. Each configuration consists of at least one ATF for an aircraft station. An example is shown in Table 2 below.

Table 2: A List of Possible ATF Configurations for F-16A [59]

Config Code*	9	8	7	6	5	4	3	2	1
AA1	M	M	M	-	ECM	-	M	M	M
AA2	M	M	M	370	300	370	M	M	M
AA3	M	M	M	-	-	-	M	M	M
AG1	M	-	A	A	300	A	A	-	M
AG2	M	-	A	370	ECM	370	A	-	M
AG3	-	-	B	B	ECM	B	B	-	-
AG4	M	-	C	-	300	A	A	-	M
AG5	M	-	C	370	300	370	C	-	M
AG6	M	M	B	B	300	A	A	M	M

A - 3 MK82'S ON A TER
B - MK84
C - 2 AGM-65'S ON A LAU-88 LAUNCHER
M - AIM-9L MISSILE ON MISSILE LAUNCHER
370 - FUEL TANK ON STATIONS 4 & 6
300 - FUEL TANK ON STATION 5

* ALL CONFIGURATIONS WILL CARRY CHAFF/ FLARE MODULES AND THE 20 MM GUN

A function by itself may consist of many tasks, and a task may consist of subtasks, etc. Thus, a plan with a set of configuration forms a complicated set combinatorical problem [60,61,62,63]. Mathematically, if we define a plan P, a configuration code C, the aircraft turnaround functions ATF, and the aircraft stations L, then P can be described as:

$$P = f(C: ATF: L) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Note that C consists of a single element C = {c = 1). That is, one configuration code is allowed in a plan:

$$ATF = \{a_n: n = 1,2,\ldots,N\} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

where a_n is ATF, N is the total number of functions in a configuration. For example in Table 2, the configuration code: ATF = AA1 = $\{a_1=a_2=a_3=a_7=a_8=a_9=$"M", $a_5=$"ECM"$\}$.

$$L = \{s_j: s_j \rightarrow j, j = 1,2,\ldots J\} \quad \ldots\ldots\ldots (3)$$

where s_j is an integer variable that takes on the value of j, for j=1, ... J, and J is the number of stations in the aircraft. For F16A configuration shown in Figure 1:

$$L = \{1, 2, 3, \ldots 9\}$$

With such a representation, an ATF plan can be described uniquely by a single or a combination of three word grammar. For example:

$$P \rightarrow (AA1: (M,3), (M,7), (ECM,5))$$

is a plan whose job configuration code requires loading AIM-9L Missiles on stations 3 and 7 and ECM pods on the centerline pylon on station 5.

In a particular configuration code, there are various ways to choose a plan. Let's take a configuration code AG2, say. The constraint which binds the set ATF to the set L defines the permutations. This scenario is shown in Table 2. The initial set match from ATF to L is as follows:

$$\boldsymbol{P} \rightarrow (\boldsymbol{AG2}:\ (M(1), A(3), 370(4), ECM(5), 370(6), A(7), M(9))$$

Thus, there are currently 7 ATF to L requests. Since r subsets can be chosen in any random or arbitrary order from P, the number in the possible universe of choices from configuration code AG2 is given by:

$$\binom{7}{r} = {}_7C_r = \frac{7!}{r!(7-r)!} \qquad \ldots\ldots\ldots\ldots\ldots\ldots (3)$$

The total plan state space generated by equation (3) defines a plan explosion. Two unique cases are: (a) for r = 7 a single plan consisting of ATF on all stations defined by AG2 must be performed; and (b) for r = 1 each plan calls for 7 unique and independent ATF on each station.

In general, for an aircraft whose structural profile is defined by equation (1) with a plan explosion vector defined by:

$$\Lambda = \begin{bmatrix} \binom{X_1}{r_1} \\ \binom{X_2}{r_2} \\ \vdots \\ \binom{X_c}{r_c} \end{bmatrix}$$

Where X_i is the total ATF to station requests in configuration code

profile i, and r_i is the possible combination request subset by the user. Thus, the total number of plans that can be generated for the aircraft is defined by y, where:

$$y = \sum_{i=1}^{c} \binom{X_i}{r_i} = \sum_{i-1}^{c} \frac{X_i!\,(r_i^{-1})!}{(X_i - r_i)!} \quad \ldots\ldots\ldots (5)$$

4.3 A Task Oriented Planner for ATF

In the world of aircraft turnaround functions, each function may consist of a single or many tasks. Each of these tasks may be subdivided into many other subtasks. Take the function A (Mk-82) for example. The subtasks are:

1. Install impulse cartridges.
2. Position and load munitions.
3. Install rack safety pin.
4. Tighten sway braces and position ejector feet against munitions.
5. Adjust and cut arming wire/fin release wire.
6. Remove fuse safety devices/pins.
7. Remove fin release band safety pin.

Each of these tasks may have micro subtasks. Associated with each of these tasks are resources (known as crews) and other constraints such as hazardous environments (chemical or biological warfare), or in a normal environment (perhaps peace time training).

An intelligent planner should be able to use the information about an ATF and plan opportunistically to achieve the goal. In this effort, we have developed a knowledge-based Task Oriented

Planner (TOP). The TOP environment utilizes the principle of Open Programming System (OPS) architecture. An OPS concept is an extension of a blackboard problem solving architecture [64] to include dynamic context focusing. In TOP, the planner is developed under the ARTFUL™ library, the scheduler is developed using a NEXPERT™ expert system shell, and the interface between the planner and scheduler is under a complied DBASE III™ Plus environment. In the TOP domain, knowledge about the world of aircraft turnaround configuration is prepared by the planner. The plan for a particular ATF is passed to the scheduler who prepares a detailed task assignment and time-window scheduler. Similar to the blackboard architecture, information migrates between the planner and the scheduler as intracellular units based on the levels of abstractions set up by the planner.

In the planning mode, TOP allows the user to interact with menus which are transparently defined. A menu declaration is driven by DECLARE():

```
DECLARE cue[menu_size ], msgs[ menu_size ], udfs[menu_size]

cues[ 1 ] = " Aircraft "
cues[ 2 ] = "Config "              && program's light-bare menu
cues[ 3 ] = " T. A. F. "
cues[ 4 ] = " Function Tasks "
cues[ 5 ] = " Calculate "
cues[ 6 ] = " Utilities "
msgs[ 1 ] = " Specific Aircraft information "
msgs[ 2 ] = " All possible Configuration for Planning "
msgs[ 3 ] = " Turnaround aircraft functions that make up
            configurations "
msgs[ 4 ] = " Tasks that make up functions "
msgs[ 5 ] = " Update totals after adding or changing functions
            times  "
msgs[ 6 ] = " Program maintenance and configuration "
udfs[ 1 ] = " THE_USUAL ('AIRCRAFT') "
udfs[ 2 ] = " THE_USUAL ('CONFIF') "
```

```
udfs[ 3 ] = " THE_USUAL ('FUNCTION') "
udfs[ 4 ] = " THE_USUAL (FUNCTASK') "
udfs[ 5 ] = " func_calc () "
udfs[ 6 ] = " UTIL_MENU  () "
```

A menu dictionary can be opened for information on a particular menu by the program:

```
IF  ! CHK_DICT(dict_file)
      SET CURSOR ON
      QUIT
ENDFILE
```

And the start up screen is controlled by the pseudo code:

```
DESKTOP()                              && startup screen
help_code = "INTRO"                    && Intro help screen
choice = 1
DO WHILE .T.                           && Here comes the program...
   help_code = "MAINMENU"
   DO_IT( choice, cues, msgs, udfs )
   IF VERIFY ( "Leave this program and return to DOS" )
       EXIT
   ENDIF
ENDDO                                  && ... There goes the program

SIGN_OFF( exe_name, exe_desc, logo_line )
```

Information on a plan is displayed on a plan window. This enables the user to interactively validate an existing configuration code or replan a new one. Each plan is written into a dBASE database file known as JOBLIST.DBF. An example plan for configuration code AA2 is shown on Table 3 below:

Configcode	°	AA2
Ac ID	°	
Station9	°	M
Station8	°	M
Station7	°	M
Station6	°	370
Station5	°	300
Station4	°	370
Station3	°	M
Station2	°	M
Station1	°	M

Table 3: An Example Plan for Configuration Code AA2

There are various options in the plan environment that allow the user to edit or change a plan structure. Some of these options are:

Add

This option allows you to add new records to the current database. The current database name is displayed in the lower left hand corner. This main menu option that you choose to get to the submenu should indicate the database to which you will be adding.

Browse

This option will allow you to view without being able to change a record in the current database. This screen allows you to view a complete record on one line of the screen. Use the cntl-cursor keys to pan over if all the fields are not visible on the screen.

Calc

This option allows you to sum all of the numeric fields in the current database. It can be used to get a total of some field to check or verify data entry.

Dup

This option is for duplicating the existing record when doing data entry. It will save a lot of key strokes if most of the data being entered has repeating fields.

Edit

This option allows for the changing of any of the fields in the currently displayed record. Press ctrl-W to save or press enter on the last field in the record to save the changes made to the record.

Find

This option allows you to search the database for a specific record based on the current index key. It will prompt you for the correct data and format before beginning the search.

List

This option will list and run any predefined reports that are in the system. The reports need to be defined in the R&R format and are not currently supported in this release of the software, but is included for future upgrade capability.

Query

This option will let the user generate a query based on any of the fields of the current database and will display on the records that make the query true. This can be used to do "what if" analysis on a specific database if desired.

Furthermore, the planner uses information task time developed by Battelle Memorial Institute [4] to predict likely time estimates to complete an ATF. The time is calculated by the formula:

$$Totime_j = t_j s_j f_j d_j \quad \ldots\ldots\ldots\ldots\ldots\ldots (6)$$

where t_j = time per task occurrence per personnel used for task j; f_j is frequency of task , during a job in ATF; s_j is the number of personnel used in task j; and:

$$d_j = \begin{cases} 1 & \text{if ATF is a nonhazardous environment} \\ > 1 & \text{otherwise} \end{cases}$$

The total time to complete a job in ATF is:

$$Jobtime_i = \sum_{j=1}^{Task} totime_j \, v_j \quad \ldots\ldots\ldots\ldots\ldots\ldots (7)$$

$$where\ V_j = \begin{cases} 1 & if\ task_j\ is\ needed\ for\ job\ i \\ o\ , & otherwise \end{cases}$$

Exhibit-1 below shows a program within the planner that accomplishes the time prediction model above:

Exhibit-1

```
function func_calc
* calculate total times for all tasks
* hazardous and non hazardous

select functask
goto top
do while .not. eof()
   replace nhaz_time with tj*sj*fj
   if dj>1 replace haz_time with nhaz_time*dj
else   skip
enddo

* sum time for  A = load tur
go top
sum haz_time for funccode = "A" to htot
sum nhaz_time for funccode = "A" to nhtot

* sum times for ECM = Electronic counter measure select functask
sum haz_time for funccode = "ECM" to htot
sum nhaz_time for funccod = "ECM" to nhtot

select function
go top
seek "ECM"
replace time2 with htot
replace time1 with nhtot

* sum time for C = Preloaded lau-88
select functask
sum haz_time for funccode = "C" to htot
sum nhaz_time for funccode = "C" to nhtot

* sum times for CUR = cursory inspection
select functask
sum haz_time for funccode = "CUR" to htot
```

```
sum nhaz_time for funccode = "CUR" to nhtot

select function
go top
seek "CUR"
replace time2 with htot
replace time1 with nhtot
pop_msg("Finished calculating .. Thank you")
return 1
```

The user can use the program above by interfacing with the screen menu options through the selection of Functions Tasks and Calculate Options. These are described below:

Functions Tasks

This option allows you to keep track of specific function tasks. When chosen it presents you with the standard submenu options and a screen of tasks specific data. This option is used to calculate the total time for each specific function task.

Calculate

This function updates the system after changes have been made to the fields of any of the databases. It must be used before running the scheduler if any changes have been made to the system. It exports data to the scheduler database.

Note that it is these predicted times that are passed into JOBLIST.dbf to be used for scheduling.

4.4 Summary

This chapter has presented TOP (a Task Oriented Planner) for the world of aircraft turnaround functions. TOP architectural design which interactively creates a plan based on contexts (user needs), is conceptualized to minimize plan storage. In addition to

this advantage, TOP allows for:

(1) Implicit replanning through window and screen functions.

(2) Prediction model for task and job processing times.

(3) For each ATF configuration code, TOP can create a dynamic partial plan by reviewing the current active plan and modifying it interactively according to the user's request.

(4) The plan interaction with the user makes it possible to generate and maintain feasible configuration codes.

CHAPTER 5

KNOWLEDGE-BASED SCHEDULING FOR AIRCRAFT TURNAROUND FUNCTIONS

5.1 Introduction

This report is concerned with the problem of scheduling turnaround function tasks using an expert system methodology. The scheduling problem can be stated as follows: find a sequence of turnaround functions that will attain the required turnaround function goal in the "shortest possible" time. The problem is difficult because the tree of possible sequences can be huge. Major emphasis has generally been placed on heuristic methods to make the problem tractable.

In the aircraft turnaround domain, the scheduling of the resources over time must be coordinated with detailed planning of the separate operations. The objective is first to find a solution that satisfies the constraint. In the current model, scheduling requires the identification of the aircraft (in this case, F16), the personnel available (human and robots), the tasks to be performed and the constraints in sequencing the tasks.

In TOP, we have an imbedded scheduler (see Figure 5) which allows jobs to be scheduled under (1) resource constraints, and (2) in parallel. The scheduler presented is generic in that it allows for either the robot or human to be used in the schedule. Further, the scheduler is plan dependent. However, its architecture and formalism are domain independent. For example, with a defined data structure similar to JOBLIST.dbf, the scheduler can still be used,

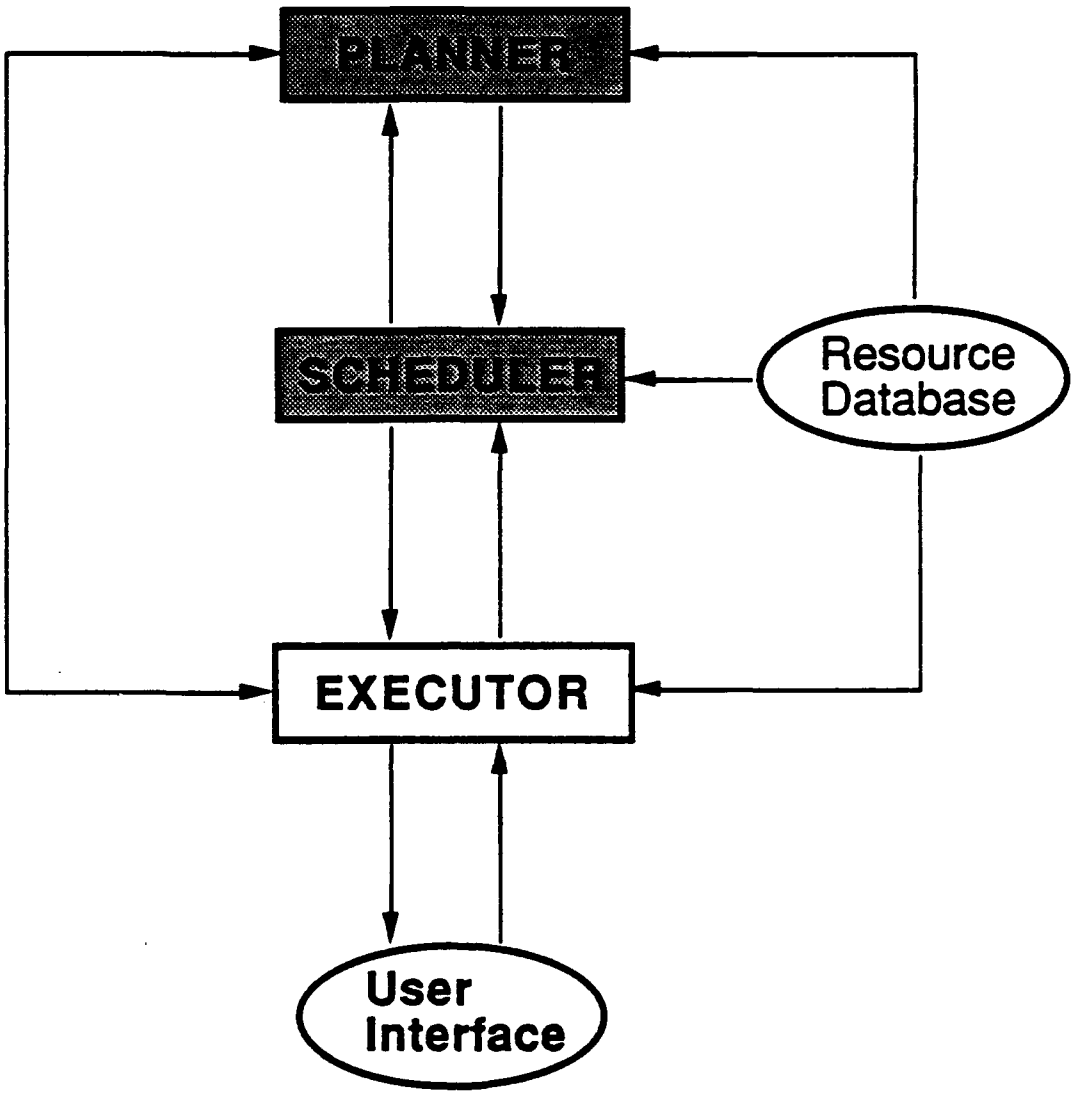


Figure 6: TOP Architecture

whether it is in a manufacturing - or project management - environment, respectively.

As shown in Figure 6, job scheduling takes place upon the completion of an ATF plan of the jobs in JOBLIST database under resource constraints. The TOP scheduler is an expert system written in NEXTPERT™ environment. The scheduler starts by reading the plan database as initial NEXPERT facts. The knowledge processing is triggered by the EXECUTOR after verifying that the initial conditions required to process a job have been satisfied. Within the scheduler, the EXECUTOR implicitly assigns jobs to ATF crews. This is accomplished by using meta rules which dynamically create nodes for each job execution, "look into" job agendas, and prioritized the job processing sequence based on the scheduling heuristic to be discussed. At the end of a schedule session with NEXPERT, the results are written into dBASE data files.

5.2 Schedule Generation

TOP generates a schedule by reasoning about a known plan. The plan generated by the planner consists of unscheduled jobs with tagged expected processing time for the jobs. The scheduler reads the plan file as its input and at the same time reads the resource file (see Figure 7). The scheduling of each job in time-phase starts after the scheduler has adaptively posted a constraint on each job based on the resource needs. Constraint posting and resolution are set-up under NEXPERT objects as dynamic nodes rules.

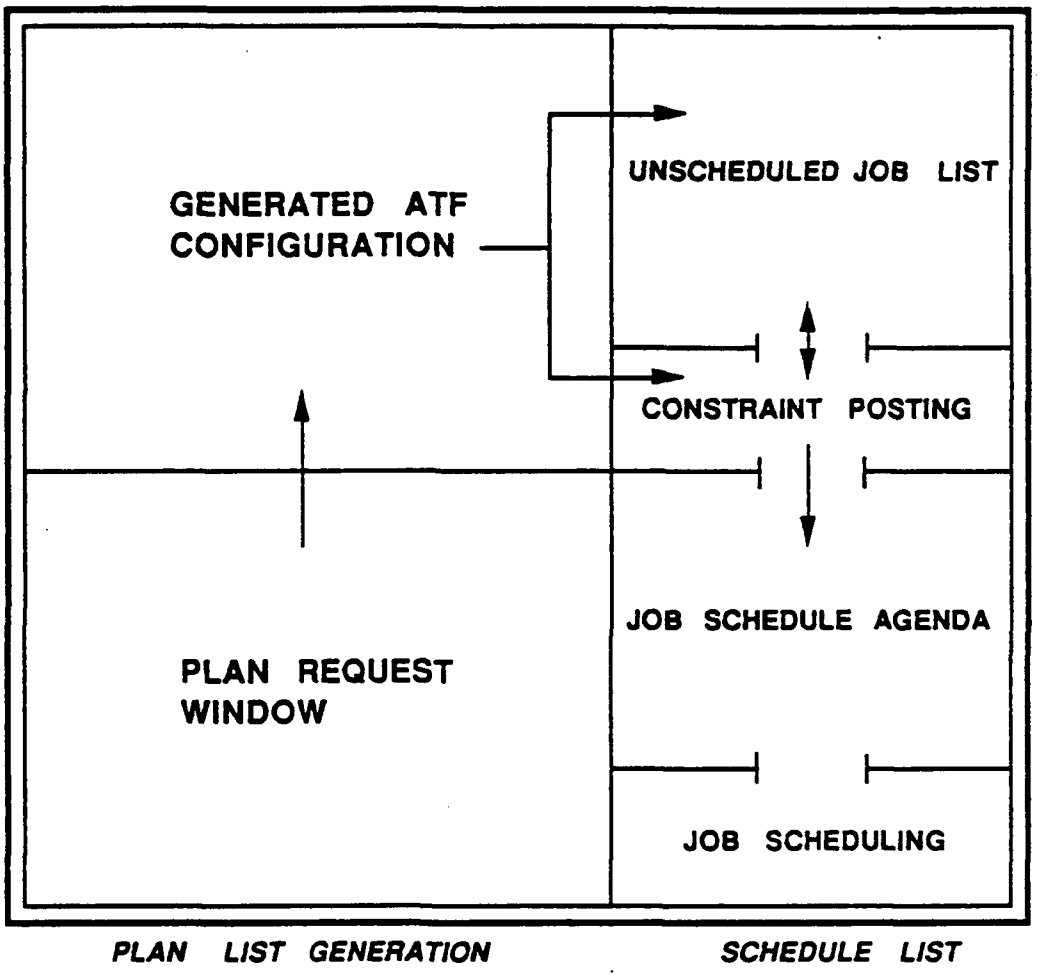


Figure 7: An Architecture for Plan-to-Schedule Generation

The TOP scheduler has 14 such rules that control dynamic job scheduling policies. These rules are set up as "a working group." If, say, a job requires three weapon crewmen, the dynamic variable "wpg3" will be set to a true value. The inference engine will use this truth value to dynamically create a working node for each weapon crewman. With this, it is possible to keep track of each personnel's status (idle or busy) during a particular instance. (Exhibit-2 shows a sample of dynamic node creation).

Exhibit-2 Sample Dynamic Node Creation In

Hypothesis	Variable	Group
nodetype1	wpg3	wp1,wp2,wp3
nodetype2	wpg2a	wp1,wp2
nodetype3	wpg2b	wp1,wp3
nodetype4	wpg2c	wp2,wp3
nodetype5	wpg1a	wp1
nodetype6	wpg1b	wp2
nodetype7	wpg1c	wp3
nodetype8	wpra	wp1,rb
nodetype9	wprb	wp2,rb
nodetype10	wprc	wp3,rb
nodetype11	cha	ch1
nodetype12	chb	ch2
nodetype13	chg	ch1,ch2
nodetype14	rb	rb

In addition, a new object node will be created whenever the job or subtask has been assigned. The newly created node, "NEWJOB" inherits the property of a job class which includes job-name, subtask, station, start-time, endtime, and personnel.

The "SEQ" field in Joblist.dbf controls the dynamic sequence of creating subtask nodes during knowledge processing. For instance, "Load Aim-9L" which has three subtasks, preloading, load and postloading, will have a "SEQ" value = 3. With a "SEQ" value

of 3, only the "preloading" node is created initially and "SEQ" will be updated to 2. With a "SEQ" value of 2, the "loading" node is created and "SEQ" becomes 1. After the last subtask, which has the "SEQ" value of 1, "postloading" has been assigned. The whole job "Load Aim-9L Missile" is complete and "job-list assigned" variable will be set to true.

In Exhibit-2, nodetype(i) is the node type i created for a schedule profile. The variables are binary values which allow for a hypothesis to be true or false. The "group" column consists of ATF crews. For example, in nodetype1, the weapon group consists of three personnel: wp1, wp2, wp3, respectively. Thus, in nodetype1 creation, the rule: IF wpg3 = "True," then load weapon with three crew members. Similarly rb is for robot, ch1 and ch2 for crew chief 1 and 2. Wpg2a, wpg2b, wpg2c are possible combinations of weapon groups (e.g.; a for 1 and 2, b for 1 and 3 and c for 2 and 3). These notations can be changed by the user.

At the completion of dynamic constraint posting and the resolution phase, a temporal job sequence is generated and placed in a job sequence agenda (See Figure 7) based on availability of resources needed for the job execution. Figure 8 shows an example job plan agenda with its corresponding plan graph. A plan graph consists of all jobs required for the success of an ATF configuration code. For a demonstration, G is the goal state which is equivalent to a desired ATF. The schedule is constructed from the job sequence agenda by time phasing a job completion time based on session resource availability.

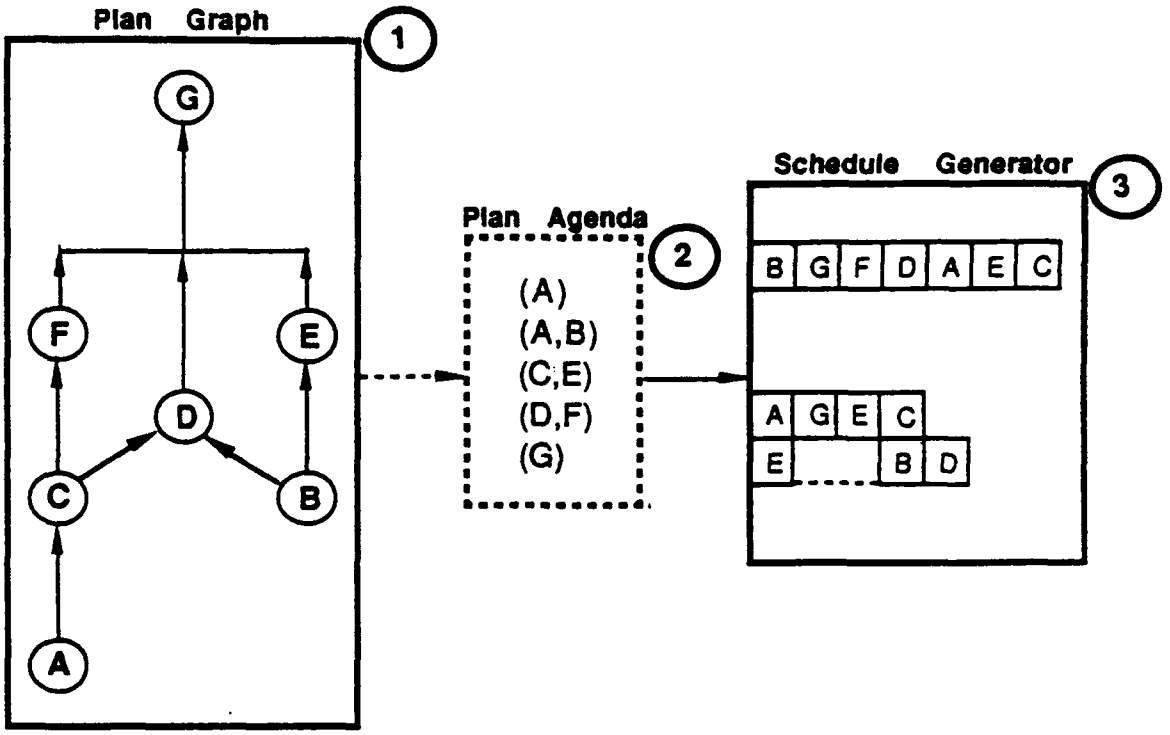


Figure 8: An Example Partial Schedule Generator From A Plan Graph

Note:

1. Let G = goal state => desired ATF.
   G = (D ^ E ^ F) means that G is realized if all jobs (D, E, and F) have been executed successfully. For example, D could be loading AIM-9L, etc.. A, B, C are subtasks. Thus, tasks B and C must have been executed to complete D. And, to accomplish C, A must be done first. The TOP program uses both backward and forward reasoning in the NEXPERT™ domain to realize this.

2. This is a possible agenda list representation in TOP. Many agenda structure can be realized. As it is, a plan agenda has no conflict among its elements.

3. This shows an example schedule profile generated from the plan agenda by NEXPERT™. When resources are used during a schedule, a schedule profile may be different.

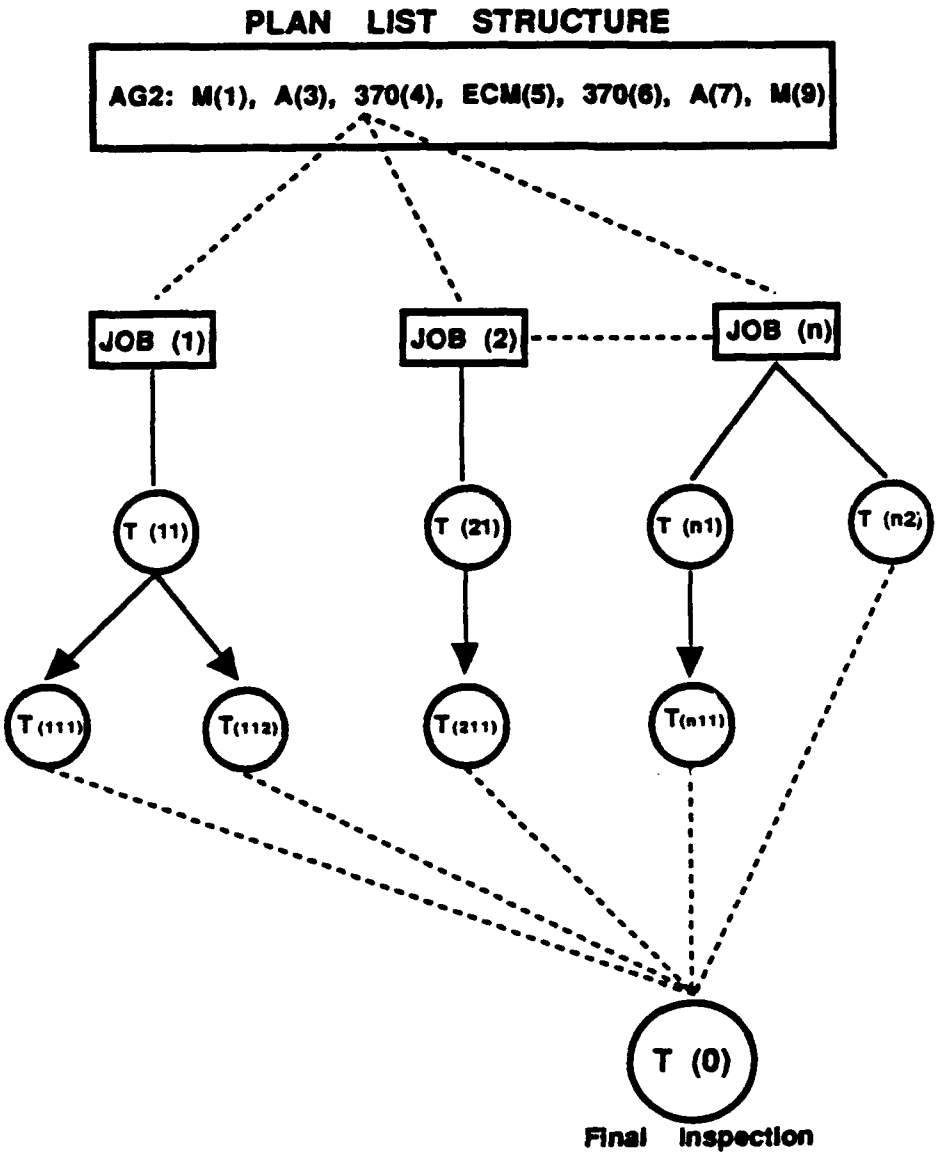
In TOP, the constraint posting phase defines a task assignment. It could be one-to-one assignment or many-to-one assignment. A one-to-one job assignment allocates a crew or a single personnel to an ATF task. Similarly, many tasks can be assigned to a simple crew, or reversely, many personnel from different crews can be assigned many tasks. In any kind of assignment, a job completion is the subgoal. TOP handles this reasoning via a top-down strategy as shown in Figure 9. This is an example AG2 configuration code whose ATF is to load three Mk82 in station 3 only. Job(1), Job(2), ... Job(n) are subtasks required for the success of this ATF. The TOP Scheduler will systematically search its knowledge base to find all the subtasks needed for the job completion. The final task denoted T(0) signifies a completion of an ATF in this case. Note that the completion time of T(0) is the maximum time taken by one of the jobs to be completed. In the next session, the process of accomplishing the scheduling profile in TOP is discussed.

5.3 TOP Heuristic Scheduling Algorithm

The scheduling procedure in TOP is based on a heuristic algorithm which is representative of an unstructured environment. The scheduling algorithm is plan dependent. Conceptually, the scheduler looks at the plan environment as follows:

$$\boldsymbol{T} = T_1, T_2, \ldots, T_n \qquad (8)$$

and a set of resources:



PLAN LIST DEFINITION

AG2: Is configuration code
= ICT plan.

J(S): Job to be performed
on station S.
e.g. A(3) is load 3 MK82's
on station 3.

JOB DEFINITION

Job (i) : Job i
T(i..) : Task or subtask for job i.
T(0) : Final check.
E.g. For A(3) plan
J(1) = TER prepration
J(2) =Loading, etc.

Figure 9: A Tree-like Task System From TOP

$$\boldsymbol{A} = A_1, A_2, \ldots, A_m \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (9)$$

$$\textit{with}\ \ A \neq \phi, T \neq \phi$$

and **A** w **T**, where w implies that **A** is not necessary less than **T**.

If there exists a set of assigned resources defined by:

$$\boldsymbol{B} = (A_i \bigcup A_j), i \neq j,$$

a set of $\mathbf{A}_i$ can be humans, $\mathbf{A}_j$ can be robots, or generically, agents with different skill levels. The plan problem is represented as a context mechanism primitive defined as:

$$\boldsymbol{P}\text{: } \{\exists z_{,k} \mid Z \in B\ Z \Rightarrow T_k\} \ldots\ldots\ldots\ldots\ldots\ldots (10)$$

Equation 3 says that in the context plan P, there exist some resources(s) Z in the available resource pool B such that the imminent set of tasks T_k can be assigned to Z.

In the plan synthesis, a concept is declared abstractly such that a condition that needs task execution can be instatiated with a transparent plan. Thus, accordingly, (See, e.g.; Sacerdoti [23]) a concept can become a context during instatiation. This makes it easy for replanning and scheduling when necessary.

Once the planning has been achieved, the scheduling phase is activated. The scheduling process in NEXPERT™ is by context mechanism (as in planning). A context mechanism creates a hierarchy of knowledge about a job by instant tracking of agendas using both forward and backward propagation. This hierarchy of knowledge provides a domain for constraint posting and constraint satisfaction checks in the world of turnaround function scheduling.

The scheduling problem can be formulated as follows: given **N**

tasks, each of which requires a certain amount of effort from a labor crew (not necessarily all at once), how should one schedule the tasks to obtain a total work load balance and work cooperation that "minimizes" a job completion time?

The following definitions are used throughout the discussion:

M	number of resources available;
T	a set of tasks;
T_i	a member in R referred to as resource type i;
R	a set of resources;
R_j	a member in R referred to as resource type j;
$\cup$	a disjunction (OR);
$\cap$	a conjunction (AND);
P	a plan or a sequence of proposed action;
$\exists$	an existential quantifier, "there exist;"
$\forall$	a universal quantifier, "for all;"
t_i	the processing time of task i;
ϵ	belong to;
k,i,j	indexes or counter notations;
n_1	a set of scheduled tasks;
n_2	a set of unscheduled tasks (or waiting tasks);
<—>	parallel logical notation; for example i <—> j means that i is parallel to j;
N	number of tasks.

Heuristically, the scheduling problem is as follows:

DO: WHILE < PLAN P $\neq$ 0 >

1. Select the plan with an immediate goal based on plan priority. Initialize job completion time F(P) = 0, and schedule clock, S=0.

2. **IF** the cardinality of **R** ≥ cardinality of **T** (i.e., if **M** ≥ **N**) then schedule all tasks simultaneously. Let $n_1 = n_2 = 0$; GO TO step 10.

3. **Else**
Select a task with available resources and store in n_1 and tasks without resources into n_2 (See, e.g.; Figure 10). Set the completion time of all jobs in n_2 to a very large value to indicate their low priorities. That is, $t_j = \infty$, for all $j \in n_2$.

4. Schedule all tasks in n_1 and update their completion times.

5. Select tasks in n_2 for processing based on the most available resources i.e.;

$$\exists_k, \exists_x (k \epsilon n_2, x \epsilon n_1; k \epsilon \lceil \forall_x \min \{t_x\} \rceil) \ldots\ldots\ldots\ldots (11)$$

Equation (11) states that the tasks in n_1 with the minimum processing time will release resources for unscheduled tasks k currently in n_2.

6. Update the schedule clock to the current end of event (first task completion) and start a new schedule from n_2. That is:

$$S = S + Min\ \{t_x\}, (see\ step\ 5) \ldots\ldots\ldots\ldots\ldots\ldots (12)$$

7. Remove x from n_1 since this task has been completed; add k to n_1 to update a list of task execution in progress; and update n_2 by deleting k which is now scheduled. That is:

$$n_1 = n_1 + k \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (13)$$

$$n_2 = n_2 - k \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (14)$$

8. If all jobs for the current plan **P** have been scheduled (i.e.; $n_1 = N$ and $n_2 = 0$) then go to step 10.

9. **Else** go to step 5.

10. Calculate the job completion time for plan P. This is given by:

$$F(P) = MAX \begin{cases} \forall_{i \epsilon N} \{\max(t_i)\}, \ if\ n_2 = 0, \ see\ step\ 2 \\ \forall_{i \epsilon n_1} \min(t_i) + Max\ \{\exists_{j \epsilon n_2} [j \epsilon i \bigcup_{j=1} | j \rightarrow \{\min(t_j)\}]\} \end{cases} \cdots (15)$$

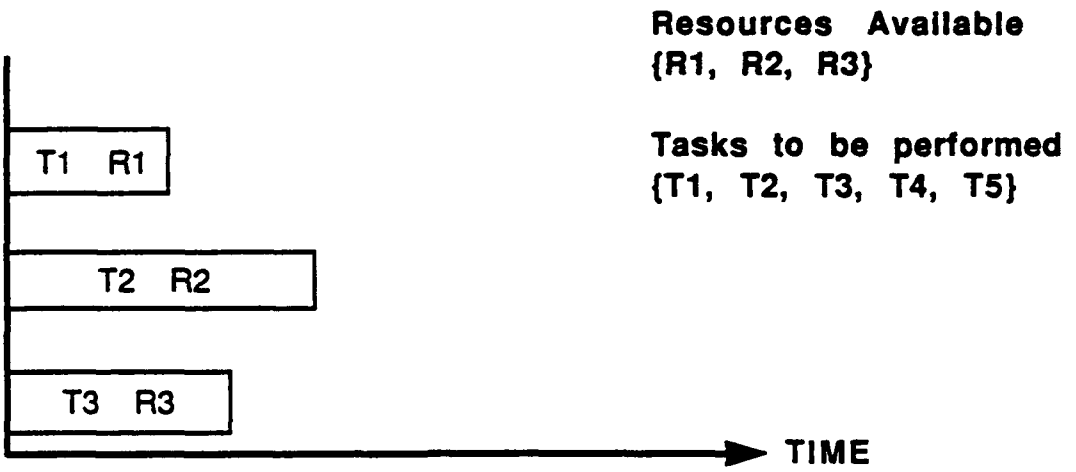


Figure 10A: Beginning Schedule

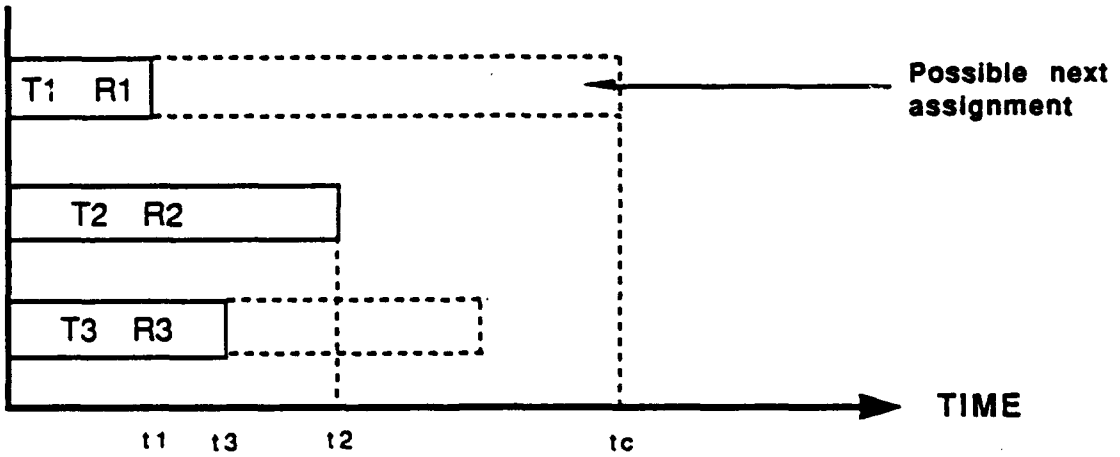


Figure 10B: A Schedule with tasks, T1 and T3, and completed at times t1 < t3. Tasks T4 and T5 are to be scheduled next t_c = potential job completion time.

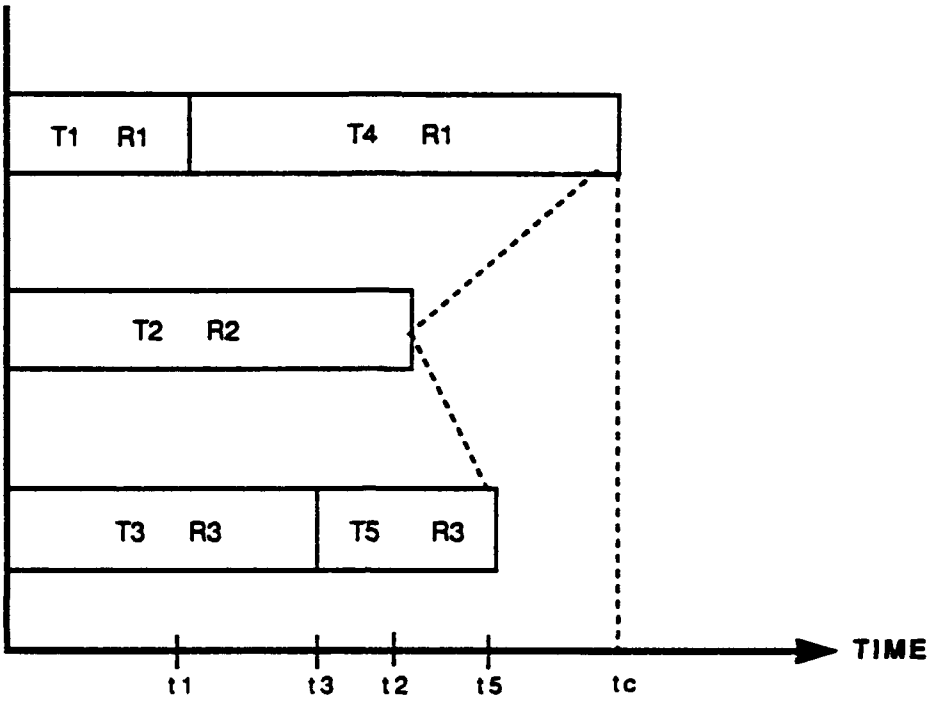


Figure 10C: A schedule showing a dynamic availability of resource R2.

R2 can be deployed in any of the two scenarios in Figures 10A and 10B.

Equation 15 states that if resources are available at the beginning of all tasks as defined in step 2, then F(P) is the time to complete the longest task. Otherwise, F(P) is the addition of the start times of all active jobs (defined by minimum time of all tasks already completed in n_1,) and the expected maximum processing time of all jobs in n_2 which can either be processed in parallel (i<—>j) or j is the immediate follower of i with the minimum processing time. Figures 10-12 are used to illustrate the above concepts.

Note that R_2 and R_3 can be robots while R_1 can be human. As shown in Figure 10B, tasks T_4 and T_5 are in list n_2 waiting to be scheduled. In this example, there are at least two ways to deploy the available resources (See Figure 10C). The decision to use R_2 along with R_1 in task T_4 (Figure 11A) or use R_2 along with R_3 in executing task T_5 (Figure 11B) depends on TOP conflict resolution ranking from the sequence agenda. In all cases, if we assume no inserted time delay, a minimum schedule turnaround time (see Figure 12) can be achieved. It should be noted that in the real situation, minimization of ATF time delay is a factor that is considered in selecting jobs from an agenda during a scheduling profile.

5.4 Summary

We have presented a technique for scheduling limited resource teleoperations with application to ATF. We view a teleoperation as a constraint-directed scheduling problem in an unstructured

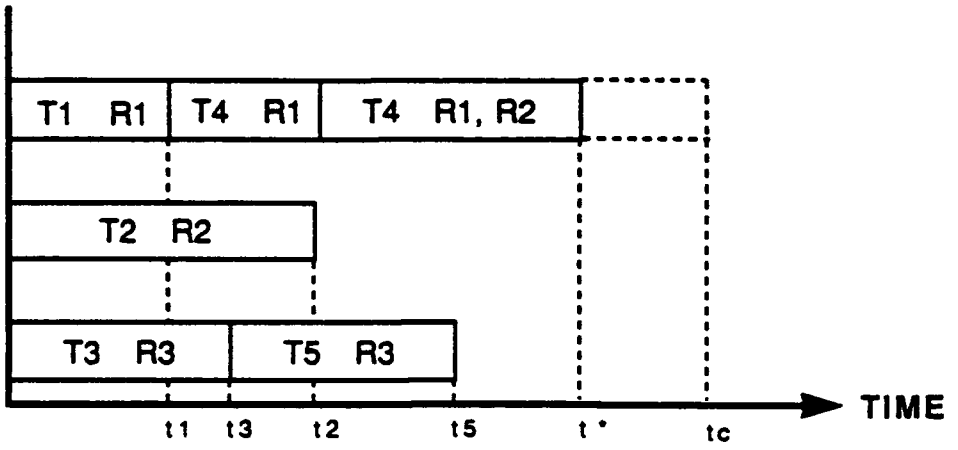


Figure 11A: The deployment of R2 in task T4 reduces t_c to t* (assume no delay time).

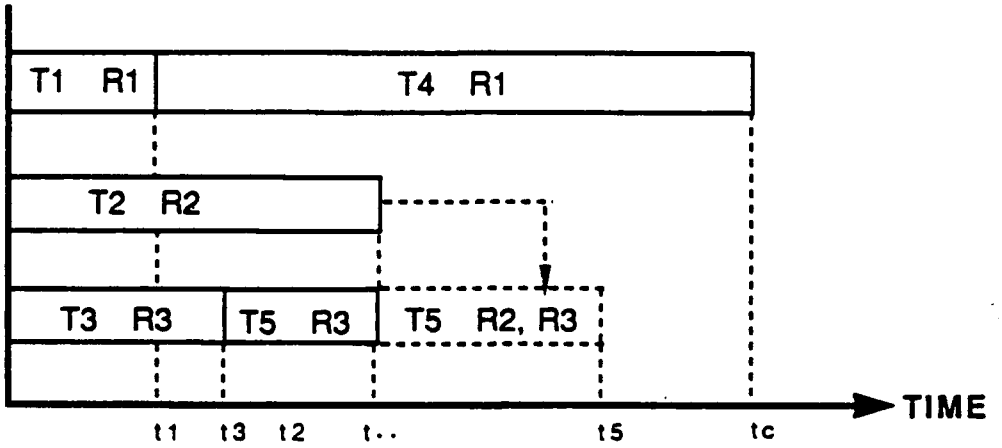


Figure 11B: The deployment of R2 in task T5 reduces t5 to t...

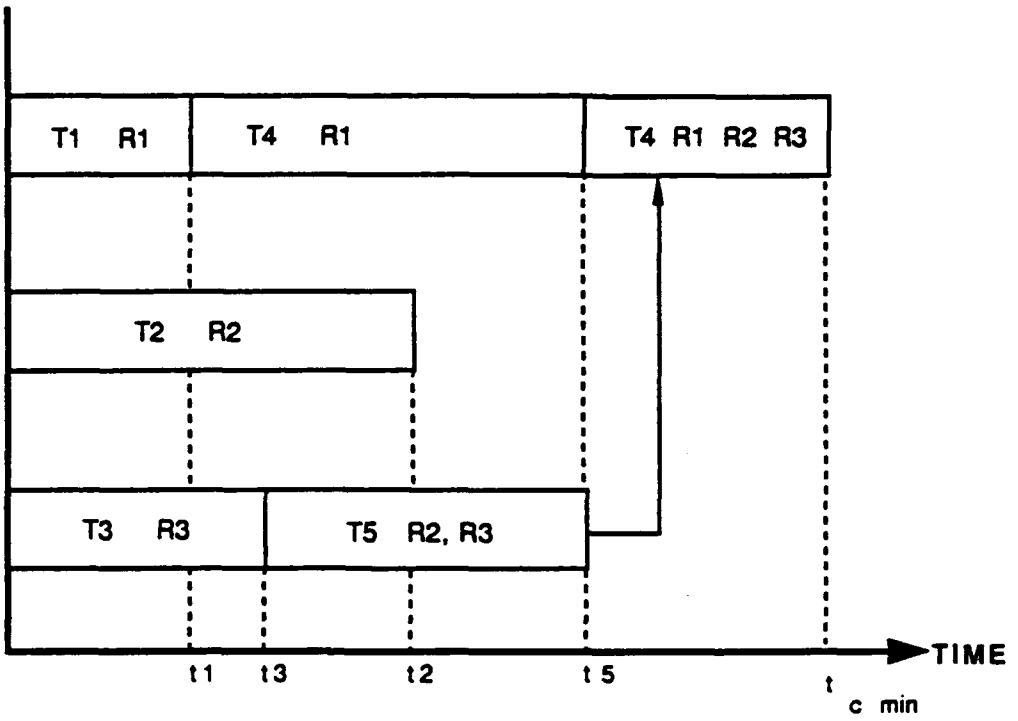


Figure 12: A possible minimum time schedule with no inserted idle time $t_{c\ min} < t_c$.

environment. The interdependence of teleoperators and their various state controls are considered as the major decision variables in the scheduling policy. Although no sets of optimization criteria have been defined for the scheduling problem [63], the algorithm is general enough to incorporate time lags

between teleoperators during task executions. Since the scheduling formalism incorporates information directly from the planner, it is easy to preempt current schedules in order to respond to plan changes. This feature of the algorithm makes it domain-independent suitable to any planning involving multiagent systems.

CHAPTER 6

MODEL APPLICATION AND VERIFICATION

6.1 Introduction

The TOP program has been successfully applied in F16-A ATF. The scheduling algorithm is written as an objected oriented code embedded in NEXPERT.

In NEXPERT, two main characteristics of objects can be distinguished: what they represent and how they should be used by the system. The structure of an object is as follows:

name
classes
subsubjects
properties

As an example, consider a turnaround function plan P1 with all the jobs to be performed: This can be represented as:

```
(     CNAME = P1
   (@CLASS = job
        (@ PROPERTIES =
      endtime          "job end time"
      id               "job ID"
      job_name         "job name"
      personnel1       "crew name:
      personnel2       "crew name"
      personnel3       "slot for floating Crew"
      starttime        "job start time"
      station          "station location"
      time1            "time for human"
      time2            "time for robot"
      timepd           "dynamic slot"
      unassigned       "logical variable to assign task"
      working-on )))
```

Thus, each job category has start and end times, dynamic slots to hold a vector of unassigned jobs, crew members, a job identification and the station number to be worked on. Generally,

NEXPERT knowledge representation allows for dynamic objects. Dynamic objects act as instances of variables which are those classes and objects. Examples of dynamic objects are "personnel13" and "timepd" slots for holding noninstantiated variables.

A very important aspect of the object organization is the notion of inheritance, that is to say, the way objects and classes can communicate values to each other. An object will typically be able to inherit a value of one of its properties from one of its classes or parent objects. For example, the door of the car will inherit the "color" of the car (parent object). All these value passing mechanisms, which represent types of default reasoning, are customizable at the lower level of granularity: the property of the object or the class. Objects also inherit functions or methods.

During a job schedule, the knowledge processing environment dynamically creates a "Node" for each personnel resource and the class attributes as discussed earlier. Dynamic node creation allows for possible job preemption, resumption and assignment of idle resources. At the end of each schedule, the system records the generated feasible schedule into a separate external dBASE file. These data remain available for later analysis.

The scheduling process uses the schedule algorithm of Chapter 5 (Section 5.2). The invocation of the algorithm uses rules to execute each schedule profile. For example, to select a plan for a schedule, Rule 85 below is fired first by setting a priority (INFACT) to the highest value possible.

```
(@RULE=    R85
     @INFCAT=100;
```

```
     (@LHS=
          (Yes (GetData)))
     (@HYPO=    (GoGet)
     (@RHS=
          (Retrieve ("c:\dbase\joblist.dbf")
(@TYPE=DBF3;@FILL=ADD;@NAME="' job_ '!id!";\
@CREATE= |job|, |timecode| '@PROPS=id, job_name,\
station, unassigned,time1,time2,starttime,\
endtime;@FIELDS="id","job_name","station",\
"unassigned","time1","time2","starttime",\
"endtime";))
          (Do   (MAX(<|job|>.id))      (n))
          (Do   (n)   (ctr)
          (Do   (1)   (wpcnt))
          (Do   (1)   (chcnt))
          (Retrieve   ("c:\dbase)upldstat.dbf")
(@TYPE=DBF3; @FILL=ADD;@NAME="' station'!id!";\
@CREATE=|station|;@PROPS=free;@FIELDS="station";\))

          (Let  (wp1.free)   (TRUE))
          (Let  (wp2.free)   (TRUE))
          (Let  (wp3.free)   (TRUE))
            (Let  (<|job|>.working_one)          (FALSE))
            (Do   (0)   (timeclock))
          (Let  (ch1.free)   (TRUE))
          (Let  (ch2.free)   (TRUE)))
```

A list of unassigned tasks can be constructed using the global rule defined by R83[1]:

```
          (@RULE=   R83
               (@LHS=
                    (=   (ctr)      (0))
                    (>   (SUM(<|job|>.unassigned))       (0))
                    (Is  (<|job|>.working_on)        (FALSE)
          )
          (@HYPO=    find_smallest_timepd1)
          (@RHS=
               (Do    (99999)   (<|job|>.timepd))))
```

The task assignments are performed asynchrously using the global rule defined by R74:

```
(@RULE=   R74
     (@LHS=
          (>    (ctr)         (0))
```

---

[1] NEXPERT defines a rule by the syntax @RULE = rule name. Thus, R83 is a rule name which in this case uses a number for easy rule identification.

```
     )
     (@HYPO=    continue_assign)
     (@RHS=
          (Reset    (continue_assign))
          (Do  ('job_'\ctr\.job_name)    (current_job.job_name))
          (Do  ('job_'\ctr\.station)     (current_job.station))
          (Do  ('job_'\ctr\.unassigned)  (current_job.unassigned))
          (Do  ('job_'\ctr\.id)     (id))
          (Do  ('ctr-1)    (ctr))))
```

Finally, the global rule to check task completion and update system status is given by:

```
(@RULE=   R79
     @INFCAT=100;
     (@LHS=
          (=    (ctr)       (0))
          (=    (SUM(<|job|>.unassigned))       (0))
     )
     (@HYPO=    done)
     (@RHS=
          (Do   (MAX(<|job|>.endtime))     (timeclock))
          (Write     ("c:\dbase\finjob.dbf")
(@TYPE=DBF3;@FILL=NEW;@PROPS=starttime,endtime,\
job_name,name,station;@FIELDS="STARTTIME",\
"ENDTIME","JOB_NAME","NAME","STATION";@ATOMS=<|personnel|>;\
))
         (Write    ("c:\dbase\jobdone.dbf")
(@TYPE=DBF3;@FILL=NEW;@PROPS=job_name,starttime,\
endtime,id,station,time1,time2,unassigned,\
personnel1,personnel2,personnel3;@FIELDS=JOB_NAME",\
"STARTTIME","ENDTIME","ID","STATION","TIME1",\
"TIME2","UNASSIGNED","PERSONNEl1","PERSONNEl2",\
"PERSONNE12";@ATOMS=<|job|;))))
```

6.2 Application Problem

The TOP program has been applied to F16-A. First the user runs the planner which is under the ARTFUL™ environment called TUR.EXE. The sample problem is the AA2 configuration code of Table 3 in Chapter 4. To run the planner, the following procedure is required of the user:

```
change to drive C:
change to directory\dbase
type TUR at the dos prompt  C:\dbase>
```

The program is self-explanatory from this stage. It allows you to add, edit, delete, browse the fields of the configuration for different turnaround functions and look at the relationship between the files used to store. If you want to/need to change any of the times, or add or delete tasks, choose the **Calculate** option on the main menu to update the entire system to changes you have made. The following are the options descriptions and how to use them. See the TOP users manual for more explanation.

The result of the Planner is database (Joblist.dbf) written to a dBase III file. This database is used as the input to TOP Scheduler. The file contains the job identification number (ID), the jobname, the station at which the job is to be performed, the expected time to perform a task under (a) human (TIME1) or (b) (TIME2). The starttime and endtime fields are slots to be completed after scheduling. The data in Table 4 show an example Joblist.dbf:

Table 4: Sample TOP Planner Output (Joblist.dbf)

Record#	ID	JOB NAME	STATION	TIME1	TIME2	STARTTIME	ENDTIME
1	1	LOAD AIM-9L MISSILE	1	5.00	4.00	0.00	0.00
2	2	UNLOAD MISSILE LAUNCHER	2	4.00	3.00	0.00	0.00
3	3	UNLOAD MISSILE LAUNCHER	3	4.00	3.00	0.00	0.00
4	4	LOAD TER RACK & 3 MK82	3	4.00	4.00	0.00	0.00
5	5	LOAD TER RACK & 3 MK82	4	4.00	4.00	0.00	0.00
6	6	UNLOAD CENTER PYLON & ECM	4	4.00	4.00	0.00	0.00
7	7	LOAD 300 & REFUEL	5	5.00	4.00	0.00	0.00
8	8	LOAD TER RACK & 3 MK82	6	4.00	4.00	0.00	0.00
9	9	UNLOAD MISSILE LAUNCHER	7	4.00	3.00	0.00	0.00
10	10	LOAD TER RACK & 3 MK82	7	4.00	4.00	0.00	0.00
11	11	UNLOAD MISSILE LAUNCHER	8	4.00	3.00	0.00	0.00

12 12	LOAD AIM-9L MISSILE	9	5.00	4.00	0.00	0.00
13 13	REFUELING	0	3.00	2.00	0.00	0.00
14 14	GUN AMMO LOADING	0	2.00	2.00	0.00	0.00
15 15	LOAD CHAFF/FLARE	0	3.00	3.00	0.00	0.00

Upon the completion of the planning session, the user is ready to enter TOP scheduling expert system resident in NEXPERT environment. To run the NEXPERT knowledge base file, following these procedures:

1. In Windows$_{TM}$, click on the Nexpert icon
2. Double clicks at Expert on the menu, then click "Load Knowledge Base"
3. Click "AFCMP.KB"
4. Double click the "True" for the value of "GetData"
5. Double click the top left corner and exit from Nexpert
6. Under Dbase, turn on the printer, and type "DO PRTLIST" to get the listings.

At the end of the scheduling session, TOP produces two output files, Jobdone.dbf and Finjob.dbf, respectively (See Figure 13). The Jobdone file gives information on random dynamic node (job) creation during a schedule generation. A typical content of jobdone file is given below in Table 5:

Table 5: Sample jobdone.dbf file

Record#	Name	JOB NAME	STATION	PERSONNEL1	PERSON	ID	START	END
1	newjob1	LOAD CHAFF/FLARE	0	ch1	ch2	-	0	3.00
2	newjob2	GUN AMMO LOADING	0	wp3	temp	-	0	2.00
3	newjob3	REFUELING	0	rb	temp	-	0	3.00
4	newjob4	LOAD AIM-9L MISSILE	9	wp1	wp2	-	0	5.00
5	newjob5	SAFETY CHECK	0	ch1	ch2	-	3	8.00
6	newjob6	LOAD AIM-9L MISSILE	1	wp3	rb	-	3	10.00
7	newjob7	LOAD TER RACK & 3 MK82	6	wp1	wp2	-	5	9.00
8	newjob8	UNLOAD CENTERLINE PYLON	5	ch1	ch2	-	8	11.00
9	newjob9	LOAD TER RACK & 3 MK82	4	wp1	wp2	-	9	13.00
10	newjob10	LOAD AIM-9L MISSILE	1	wp3	rb	-	10	15.00
11	newjob11	LOAD 300 & REFUEL	5	ch1	ch2	-	11	16.00
12	newjob12	LOAD AIM-9L MISSILE	1	wp2	wp3	-	15	20.00
13	newjob13	LOAD AIM-9L MISSILE	9	wp1	rb	-	15	20.00

14	newjob14	LOAD AIM-9L MISSILE	9	wp2	wp3	-	20	25.00
15	newjob15	UNLOAD MISSILE LAUNCHER	8	wp1	wp2	-	25	29.00
16	newjob16	UNLOAD MISSILE LAUNCHER	7	wp1	wp2	-	29	33.00
17	newjob17	LOAD TER RACK & 3 MK82	7	wp2	wp3	-	33	37.00
18	newjob18	UNLOAD MISSILE LAUNCHER	3	wp1	wp2	-	37	41.00
19	newjob19	LOAD TER RACK & 3 MK82	3	wp2	wp3	-	41	45.00
20	newjob20	UNLOAD MISSILE LAUNCHER	2	wp1	wp2	-	45	49.00

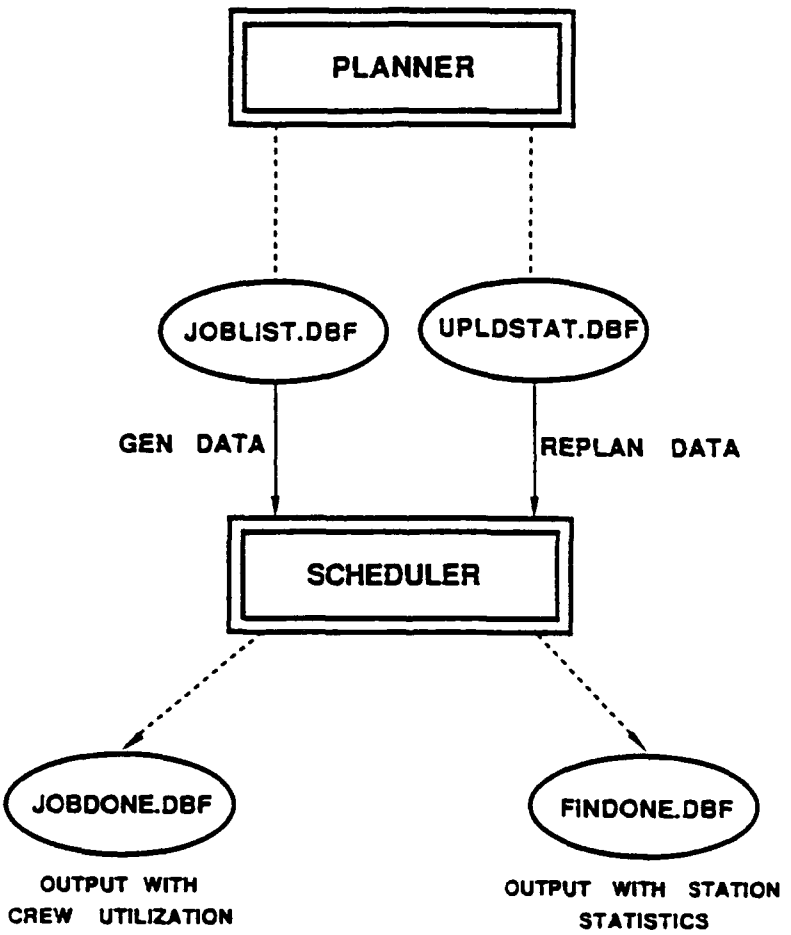


Figure 13: Output (Information) flow in TOP

The Findjob.dbf is an output with crew utilization. This output gives a complete time phase schedule for an ATF configuration code. A typical file content is given below in Table 6:

Table 6: A Complete Schedule For An ATF Configuration Code

Record#	STARTTIME	ENDTIME	JOB NAME	Name
1	0.0	3.0	LOAD CHAFF/FLARE	ch1
2	0.0	3.0	LOAD CHAFF/FLARE	ch2
3	0.0	2.0	GUN AMMO LOAD	wp3
4	0.0	3.0	REFUELING	rb
5	0.0	5.0	LOAD AIM-9L MISSILE	wp1
6	0.0	5.0	LOAD AIM-9L MISSILE	wp2
7	3.0	8.0	SAFETY CHECK	ch1
8	3.0	8.0	SAFETY CHECK	ch2
9	5.0	10.0	LOAD AIM-9L MISSILE	wp3
10	5.0	10.0	LOAD AIM-9L MISSILE	rb
11	5.0	9.0	LOAD TER RACK & 3 MK82	wp1
12	5.0	9.0	LOAD TER RACK & 3 MK82	wp2
13	8.0	11.0	UNLOAD CENTERLINE PYLON & ECM	ch1
14	8.0	11.0	UNLOAD CENTERLINE PYLON & ECM	ch2
15	9.0	13.0	LOAD TER RACK & 3 MK82	wp1
16	9.0	13.0	LOAD TER RACK & 3 MK82	wp2
17	10.0	15.0	LOAD AIM-9L MISSILE	wp3
18	10.0	15.0	LOAD AIM-9L MISSILE	rb
19	11.0	16.0	LOAD 300 & REFUEL	ch1
20	11.0	16.0	LOAD 300 & REFUEL	ch2
21	15.0	20.0	LOAD AIM-9L MISSILE	wp2
22	15.0	20.0	LOAD AIM-9L MISSILE	wp3
23	15.0	20.0	LOAD AIM-9L MISSILE	wp1
24	15.0	20.0	LOAD AIM-9L MISSILE	rb
25	20.0	25.0	LOAD AIM-9L MISSILE	wp2
26	20.0	25.0	LOAD AIM-9L MISSILE	wp3
27	25.0	29.0	UNLOAD MISSILE LAUNCHER	wp1
28	25.0	29.0	UNLOAD MISSILE LAUNCHER	wp2
29	25.0	29.0	UNLOAD MISSILE LAUNCHER	wp3
30	29.0	33.0	UNLOAD MISSILE LAUNCHER	wp1
31	29.0	33.0	UNLOAD MISSILE LAUNCHER	wp2
32	29.0	33.0	UNLOAD MISSILE LAUNCHER	wp3
33	33.0	37.0	LOAD TER RACK & 3 MK82	wp2
34	33.0	37.0	LOAD TER RACK & 3 MK82	wp3
35	37.0	41.0	UNLOAD MISSILE LAUNCHER	wp1

36	37.0	41.0	UNLOAD MISSILE LAUNCHER	wp2
37	37.0	41.0	UNLOAD MISSILE LAUNCHER	wp3
38	41.0	45.0	LOAD TER RACK & 3 MK82	wp2
39	41.0	45.0	LOAD TER RACK & 3 MK82	wp3
40	45.0	49.0	UNLOAD MISSILE LAUNCHER	wp1
41	45.0	49.0	UNLOAD MISSILE LAUNCHER	wp2
42	45.0	49.0	UNLOAD MISSILE LAUNCHER	wp3

In addition to the above outputs, records on subtask executions such as preloading, loading, and postloading ammunition can be viewed. Table 7 below shows sample output:

Table 7: Sample Status of Subtask Execution

Record#	STATION	SUBTASK
1	0	---
2	0	---
3	0	---
4	0	---
5	9	PRE-LOADING
6	9	PRE-LOADING
7	0	---
8	0	---
9	1	LOADING
10	1	LOADING
11	6	---
12	6	---
13	5	---
14	5	---
15	4	---
16	4	---
17	1	LOADING
18	1	LOADING
19	5	---
20	5	---
21	1	POST-LOADING
22	1	POST-LOADING
23	9	LOADING
24	9	LOADING
25	9	POST-LOADING
26	9	POST-LOADING
27	8	---
28	8	---

29	8	---
30	7	---
31	7	---
32	7	---
33	7	---
34	7	---
35	3	---
36	3	---
37	3	---
38	3	---
39	3	---
40	2	---
41	2	---
42	2	---

TOP also maintains a status report on each station during scheduling. The Upldstat.dbf is the "update statistics" file that contains dynamic flaggers of each station. Each flag (true or false) identifies which station is currently active during the schedule session. If the flag is true, automatic data keeping is maintained during the sessions. Also, if replanning should take place, the status report on the station changes accordingly. An example of file content during planning and scheduling sessions is shown in below:

Record#	ID	STATION
1	1	.F.
2	2	.F.
3	3	.F.
4	4	.T.
5	5	.F.
6	6	.T.
7	7	.F.
8	8	.F.
9	9	.F.

6.3 Summary

We have presented a prototype application of TOP to F16-A

aircraft turnaround function. As mentioned earlier, TOP planning and scheduling environment is open-ended, thus ATF for other types of aircraft is applicable. With a proper application environment definition (data base preformatting), the TOP system can respond to a new environment without knowledge degeneration. Its ability to reconfigure and replan, and heuristically schedule in TOP avoids the so called nonsolvable hard problems which are classically inherent in most scheduling problems.

CHAPTER 7

PROJECT SUMMARY

7.1 Accomplishment

Planning and scheduling in a telerobot system are two complex tasks because of the human-machine requirements that must be addressed. In a human system, these issues can well be approached from behavioral models. On the other hand, robot systems can be planned algorithmically by exploiting the available computational techniques.

A teleoperation requires direct cooperation between the agents involved in the system. Thus, a methodology to achieve such cooperation must be developed. In this project, a computational approach has been developed to assign tasks to multiagents working cooperatively in jobs that require a telerobot in the loop. Of course, this approach allows for such concepts to be applied in a nonteleoperational domain. The accomplishments under this effort are as follows:

1. In achieving human-machine planning that coexists based on the system characteristics, we have developed a planner that exploits human intentions during problem solving.
2. Considering the fact that a telerobot operates in a hostile and nonstructured environment, task scheduling should respond to environmental changes. In this regard, a general heuristic has been developed for scheduling jobs in a human-machine system. The technique is not to optimize a given scheduling criterion as in classical job -- and/or flow -- shop problems.

For a teleoperation job schedule, criteria are situation dependent. Therefore, goal achievement is emphasized with minimum expected risk to the human operator.

3. The world of ATF is characterized by unstructured input thereby making, planning and scheduling task complex. In this project, the TOP architecture developed is conceptualized as an open-ended platform decision support tool that can solve context dependent problems such as in ATF. TOP has an implicit replanning ability and can create dynamic partial schedules in response to new contexts. This ability to recognize context situations is based on the TOP knowledge base which the user can interact with to plan or replan ATF configuration codes.

4. The TOP system, although context (plan) dependent, has a scheduling algorithm that is domain independent. Thus, TOP can be applied to a variety of scheduling problems that are not well defined (i.e., do not follow a particular initial sequence).

7.2 Suggested Further Work

Although the concept of teleoperation is not new, the use of a telerobot in a more "intelligent" fashion needs a lot of research. The planning and scheduling discussions we have presented represent a subset of several works in the area of telerobotics. If we agree to look at a telerobot as a human-machine system, then the following fundamental questions are

posed as the premises of the basic research issues of the future:

1. An embedded telerobotic control, communication and command model that interactively executes the assigned ATF tasks. The current system only assigns tasks to either the human or robots and provides predictive scheduling of those tasks. A comparison of the time actually taken by these agents (telerobot and humans) will help to validate the predictive schedule time outputs.
2. An expansion of the TOP knowledge base to accommodate varieties of aircraft, detailed task elements, and other configuration codes.
3. The current TOP search strategy is too slow. The reason is that unlike most scheduling systems, the TOP scheduler tries to search through a chain of job nodes to identify: (a) job status (completed, partially completed, not assigned), (b) the crew status (idle or not), and (c) crew assignment priority (i.e.; if the crew has been dedicated for a particular task such as the weapon crew). One suggestion to this problem is to explore the use of a knowledge maintenance model which can preserve information about a schedule in a particular configuration code plan. This is what TOP is currently lacking.
4. An expansion of the TOP control strategy to the job controller and line directors level. In addition, the aircraft preparatory turnaround function procedures, involving

maintenance, inspection and on air refueling should be incorporated into TOP. Such procedures are well discussed by Chawla and Hagins[1]. With such enhancements, it will be necessary to create an interactive run time platform within TOP and the user.

References

1. Chawla, M.D. and Hagins, S.E., (May 1989). "Robotics For Flightline Servicing," National Aerospace Electronic Conference (NAECON), Dayton, Ohio.

2. Chawla, M.D. and Hagins, S.E., (Aug 1989). "Aircraft Ground Support and Robotics," Second Workshop on Military Robotic Applications, Kingston, Ontario, Canada.

3. Smith, J.L., et al., (1988). "Study Of Robotic Concepts For Aircraft Turnaround," Report # AFWAL-TR-88-3058, Vol. 1, Flight Dynamics Lab., Air Force Wright Aeronautical Lab., Air Force Systems Command, U.S. Air Force, WPAFB, Ohio.

4. Chapman, D., "Planning For Conjunctive Goals," Artificial Intelligence, Vol. 32, pp. 333-377.

5. Smith, J.L., et al., (1988). "Study of Robotic Concepts For Aircraft Turnaround." "Development Of Effective Analysis Tool For Robotic Aircraft Turnaround Concepts," Report # AFWAL-TR-88-3058, Vol. II Flight Dynamics Lab., Air Force Wright Aeronautical Lab., Air Force Systems Command, U.S. Air Force, WPAFB, Ohio.

6. Mehrez, A. and Helman, S.I., (1985). "Optimal Refueling Strategies For A Mixed-Vehicle Fleet," Naval Research Logistics, Vol. 32, pp. 315-328.

7. Sheridan, (1974). "Planning in a Hierarchy of Abstraction," Artificial Intelligence, 5, pp. 115-135.

8. Swartout, W., (Summer 1988). "DARPA Santa Cruz Workshop on Planning." AI Magazine, pp. 115-130.

9. Ntuen, C.A. and Park, E.H., (1988). "A Fuzzy Expert Simulation For Dynamic Task Allocation In A Telerobotic Environment," The Proc. of 1988 Southeastern Simulation Conference, Oct. 17-18, Orlando, FL. pp. 122-126.

10. Coiffet, P., (1981). Robot Technology, Vol. 1: Modeling and Control. Prentice Hall, Inc., Englewood Cliffs, N.J.

11. Appelt, D.E., (1980). "A Planner for Reasoning About Knowledge and Action," First Annual Nat. Conf. on AI, Stanford Univ., CA, pp. 131-133.

12. Bruce, B. and Newman, D., (1978). "Interacting Plans," Cognitive Science, 2, pp. 195-233.

13. Christiansen, A.D., (1985). "The History Of Planning Methodology: An Annotated Bibliography," SIGART Newsletter 94, pp. 44-53.

14. Corkill, D.D., (1979). "Hierarchical Planning In A Distributed Environment," Proc. of Int. Joint, Conf on AI - 79, Tokyo, Japan, pp. 168-175.

15. Decker, K.S., (1987). "Distributed Problem-Solving Techniques: A Survey," IEEE Trans. On Systems, Man, Cybernetics, SMC-17 (5), September/October, pp.729-740.

16. Newell, A.G., Shaw, J.C. and Simon, H.A., (1960). "Report on A General Problem Solving Program," Proc. Intl. Conference on Information Processing, UNESCO, Paris, pp. 256-264.

17. Fikes, R.E. and Nilsson,N.J., (1971). "STRIPS: A New Approach To The Application of Theorem Proving To Problem Solving," Artificial Intelligence, 2, pp. 189-208.

18. Siklossy, L. and Dreussi, J., (1973). "An Efficient Robot Planner Which Generates Its Own Procedure," Proc. of 3rd Intl. Joint Conf. on AI, pp. 432-430.

19. Hayes-Roth, B. and Hayes-Roth, F., (1979). "Cognitive Model of Planning," Cognitive Science, Vol. 3, pp. 275-310.

20. Fox, M.S., (1983). "Constraint Directed Search: A Case Study of Job-Shop Scheduling," Ph.D. Thesis, Carnegie Mellon University, Pittsburgh, P.A.

21. Fox, B.R. and Kempf, K.G., (March 1985). "Opportunistic Scheduling For Robotic Assembly," IEEE 1985 Intl. Conf. On Robotics and Automation, St. Louis, MO, pp. 880-889.

22. Newman, P.A. and Kempf, K.G., (Dec 1985). "Opportunistic Scheduling for Robotic Machine Tending," Second Conf. on AI Application's IEEE Computer Society, Miami Beach, FL., pp 168-175.

23. Sacerdoti, E.D., (1975). "Structure For Plans and Behavior," Ph.D. thesis Stanford University, CA.

24. Cohen, P., (1984). "Progress Report On The Theory Of Endorsements," Report # 84-15, Computer and Information Science Dept., Univ. of Massachusetts, MA.

25. Gilmore, J., et al., (1985). "PLANET: Planning and Scheduling," Report # A3896, AI Branch, Georgia Tech. Research Institute, Atlanta, GA.

26. Hammon, K.J., (1983). "Planning and Goal Interaction," Proc. of Amer. Assn. of AI, Washington, DC, pp. 148-151.

27. Hammon, K.J. "CHEF: A Model of Case-Based Planning," In Proceeds of 5th National Conf. on AI, Menlo Park, CA, pp. 267-271.

28. Wilensky, D.E., (1981). "Meta-Planning," Cognitive Sciences, 5, pp. 197-233.

29. Dorfman, P.W. and Goldstein, I.L., (1971). "Spatial and Temporal Information As Cues In a Time-Sharing Task," Journal of Applied Psychology, Vol. 55, pp. 554-558.

30. Robinson, A.E. and Wilkins, D.E., (1980). "Representing Knowledge in An Interactive Planner," First Annual Nat. Conf. on AI, Stanford Univ., pp. 148-150.

31. Licklider, J.C.R., (1960). "Man-Computer Symbiosis," IEEE Transactions On Human Factors In Electronics HFE-1 (No.1), pp. 4-11.

32. Goldstein, I.P. and Roberts, R.B., (Aug 1977) "NUDGE: A Knowledge-Based Scheduling Program," Fifth Intl. Joint Conf. on AI, Cambridge, MA, pp. 257-263.

33. Reddy, Y.V. and Fox, M.S., (1982). "KBS: An Artificial Intelligence Approach To Flexible Simulation," Tech. Report, CMU-RI-TR-82-1, CMU.

34. Stefik, M.J., (1981). "Planning With Constraints," Artificial Intelligence, 16(2) pp. 11-140.

35. Waldinger, R., (1977). "Achieving Several Goals Simultaneously," In Machine Intelligence (E.W. Elcock and D. Mitchie, Eds), New York: Halstead/Wiley.

36. Warren, D.H.D., (1974). "WARPLAN: A System for Generating Plans," Memo 76, Dept. of Computational Logic, Univ. of Edinburgh, England.

37. Garey, M.R., Graham, R.L. and Johnson, D.S., (1978). "Performance Guarantees for Scheduling Algorithms," Operations Research, Vol. 26(1), pp. 3-21.

38. Jenkins, L., (1988). "Space Telerobotic Systems: Applications and Concepts," Proceedings of Space Telerobotics Workshop, pp. 29-34.

39. Harmon, S.Y., (1988). "Dynamic Task Allocation and Execution Monitoring in Teams of Cooperating Humans and Robots," Proceedings Workshop On Human-Machine Symbiotic Systems, Oak Ridge Tennessee, December 5-6; pp. 79-98.

40. Parker, L.E. and Pin, F.G., (1987). "Dynamic Task Allocation for a Man-Machine Symbiotic Systems," Report # ORNC/TM-10397/CESAR-87/09, Oak Ridge National Lab.

41. Sheridan, T.B., (1988). "Man-Machine Communication for Symbiotic Control," Workshop On Human-Machine Symbiotic Systems (Parker, L.E. and L.R. Weisbin, Eds.), Oak Ridge, Tenn, pp. 21-35.

42. Martin, H.L. and Kuban, D.P., (1985). Teleoperated Robotics In Hostile Environments, Dearborn: SME Publications.

43. Yang, J-Y. D., et al., (1985). "An Architecture For Control and Communications In Distributed Artificial Intelligence Systems," IEEE Transactions on Systems, Man and Cybernetics, SMC -15 (3), pp. 316-326.

44. Joran, J., Meltzer, B. and Mitclue, D., (1970). "Planning and Robots in Machine Intelligence," Machine Intelligence, 5, pp. 296.

45. Klar, W., (Sep 1983). "TABS: A Knowledge-Based Time Planning System," Intl. Conf. on Networks and Electronic Office Systems, IEEE, Reading, Berkshire, England, pp. 175.

46. Smith, R.G. and Davis, R., (1981). "Frameworks For Cooperation In Distributed Problem Solving," IEEE Transactions on Systems, Man and Cybernetics, SMC-11 (1), pp. 61-70.

47. Harrison, F.W. and Pennington, J.E., (1986). Proc. of SPIE Cambridge Symposium On Optical and Optoelectronic Engineering, Cambridge, Mass, October 26-31.

48. Harrison, F.W. and Orlando, N.E., (1984). "System-Level Approach Automation," Fourth Annual UAH/UAB Robotics Conference, Huntsville, Alabama, April 26.

49. Orlando, N.E., (1983). "A System For Intelligent Teleoperation Research," AIAA Computers In Aerospace Conf., Hartford, CN.

50. Sato, T. and Hirai, S., (1987). "Motion Understanding and Structured DD Master-Slave Manipulator For Cooperative Teleoperator," Third Int. Conference on Advanced Robotics (ICAR' 87), October 13-15, Versailor, France.

51. Sato, T. and Hirai, S., (1987). "Language-Aided Robotic Teleoperation System (LARTS) for Advanced Teleoperation," IEEE Journal of Robotics and Automation, RA-3(5), pp. 476-481.

52. Hirai, S. and Sato, T., (1985). "Advanced Master-Slave Manipulator Augmented," The 15th International Symposium on Industrial Robots, Tokyo Japan, pp. 137-144.

53. Rasmussen, J., (1983). "Models of Mental Strategies In Process Plant Diagnosis," In Human Detection and Diagnosis of System Failure (M.J. Rasmussen & W.B. Rouse, Ed., Plennum Press, New York).

54. Manfred, K. and Galanter, E.H., (1958). "The Aquisition and Utilization of Information in Problem Solving," Information and Control, Volume 1, pp. 267-288.

55. Berlin, D.L.S., (1985). "SPAN: Integrating Problem Solving Tactics," 9th Int. Joint Conf. AI, Vol. 1, pp. 1047-1051.

56. Konolidge, K. and Nilsson, N.J., (Aug 1980). "Multi-Agent Planning Systems," Proc. of First Conf. of the American Assoc. for AI, pp. 230.

57. Hommertzheim, Masud, D.A. and Huffman, J., (1987). "Expert System For Military Aircraft Loading Strategies," Proc. of IIE Integrated Systems Conference, pp. 183-187.

58. Albus, J.S., Barbera, A.J. and Nagel, R.N., (1981). "Theory and Practice of Hierarchical Control," Proceedings of the Twenty-Third IEEE Computer Society International Conference.

59. Integrated Combat Turnaround Procedures, Technical Manual, T.O. 1F-16A-73-1-4, U.S. Air Force, June 29, 1988 (Change 1).

60. Hall, N.G., Rhee, K.A. and Rhee, W.T., (1988). "A Nonidentical Parallel Processor Scheduling Problem," Naval Research Logistics, Vol. 35, pp. 419-424.

61. Melkman, A.A., Helman, S.I. and Mehrez, A., (1986). "Optimal Refueling Sequence For A Mixed Fleet With Limited Refueling," Naval Research Logistics, Vol. 33, pp. 759-762.

62. Mehrez, A., Helman, S.I. and Ronem, D., (1983). "Vehicle Fleet Refueling Strategies To Maximize Operational Range," Naval Research Logistics, Vol. 30, pp. 319-342.

63. Carbonell, J.G., (1981). "Counterplanning: Model Of Planning In Real Situations," Artificial Intelligence, pp. 295-329.

64. Robinson, A., (Jan 1983). "Research in Planning," SIGART Newsletter, pp. 27-36.

BIBLIOGRAPHY

1. Atkinson, D.J., (1988). "Telerobot Task Planning and Reasoning: Introduction to JPL AI Research," Proceedings of Space Telerobotics Workshop, pp. 339-350.

2. Chang, K.H. and Wee, W.G., (Jun 1985). "A Knowledge Based Planning System For Robot Assembly," 22nd ACM/IEEE Design Automation Conf, pp. 330-336.

3. Davis, R.H. and Comacho, M., (1984). "The Application of Logic Programming To The Generation of Plans For Robots," Robotics, 2, pp. 137-146.

4. Fahlman, S.E., (1974). "A Planning For Robot Construction Tasks," Artificial Intelligence, 5(1), pp. 1-49.

5. Fikes, R.E., Hart, P.E. and Nilsson, N.J., (1972). "Learning and Executing Generalized Robot Plans," Artificial Intelligence 3(4), pp. 251-228.

6. Fikes, R.E., Hart, P.E. and Nilsson, N.J., (1972). "Executing Generalized Robot Plans," Artificial Intelligence, 3(4), pp. 251-288.

7. Fikes, R.E. and Nilsson, N.J., (1972). "Some New Directions In Robot Problem Solving," In Machine Intelligence Mectzer & D. Michie, Eds.), Vol. 7, pp. 405-430.

8. Gasser, L. and Bekey, G., (1987). "Task Allocation Among Multiple Intelligent Robots," Proceedings of the 1987 Workshop On Space Telerobotics, Vol. 1 Rodriques, G., ed.), JPL, Pasadena, CA., pp. 127-130.

9. Grant. T.J., (1985). "An Expert Fuzzy Planner for Scheduling Aircraft Repair Work," Paper presented at First International Expert Systems Meeting, London, Oct. (1-3).

10. Hayes, P.J., (1975). "A Representation for Robot Plans," Proc. of Intl. Joint Conf. on AI, USSR, pp. 104-112.

11. Kuiper, B., (1977). "Modeling Spatial Knowledge," 5th Int. Conf. On AI.-77, MIT, Cambridge, MA, Aug. 22-25, pp. 298.

12. McCallan, M.I. and Reid, L., (1982). "Plan Creation, Plan Execution and Knowledge Acquisition in a Dynamic Microworld," Intl. Journal of Man-Machine Studies, 16, pp. 89-112.

13. Pease, M.C., (1978). "An Experimental Automated Command Support System," IEEE Trans On System, Man, and Cybernetics, SMC-8(10), pp. 725-735.

14. Shin, K.G. and Epstein, M.E., (1987). "Intertask Communication In An Integrated Multirobot System," RA-3 (2), pp. 90-100.

15. Sobek, R.P., (1985). "A Robot Planning Structure Using Production Rules," Intl. Joint. Conf. on AI.

16. Stefik, M.J., (1981). "Planning and Meta-Planning," Artificial Intelligence, 16(2), pp. 141-169.

17. Tangwongsan, S. and Fu, K.S., (1979). "An Application of Learning To Robot Planning," Intl. Inl. of Computer and Info. Science, (4), pp. 626-650.

18. Tate, A., (1977). "Generating Project Networks," Proc. of the 5th Intl. Conf. on AI, MIT Cambridge, MA., pp. 888-893.

19. Vere, S.A., (1983). "Planning in Time, Windows and Duration For Activities and Goals," Pattern Analysis and Machine Intelligence PAMI-5(3), pp. 246-266.

20. Vere, S.A., (Apr 1985). "Deviser: An AI Planner for Spacecraft Operations," Aerospace America, pp. 50-53.

21. Wilhelm, W.E., (1987). "Complexity of Sequencing Tasks in Assembly Cells Attended By One or Two Robots," Naval Research Logistics, Vol. 34, pp. 721-738.

22. Wilkins,D.E., (1984). "Domain-Independence Planning: Representation and Plan Generation," Artificial Intelligence, pp. 269-301.